# BUILT-IN SELF TEST (BIST) FOR REALISTIC DELAY DEFECTS

A Thesis

by

KARTHIK PRABHU TAMILARASAN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2010

Major Subject: Computer Engineering

Built-In Self Test (BIST) for Realistic Delay Defects

# BUILT-IN SELF TEST (BIST) FOR REALISTIC DELAY DEFECTS

A Thesis

by

KARTHIK PRABHU TAMILARASAN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

| | |
|---|---|
| Co-Chairs of Committee, | Duncan Moore Hank Walker |
| | Gwan Choi |
| Committee Members, | Weiping Shi |
| Head of Department, | Costas Georghiades |

December 2010

Major Subject: Computer Engineering

# ABSTRACT

Built-In Self Test (BIST) for Realistic Delay Defects.

(December 2010)

Karthik Prabhu Tamilarasan, B.E., Anna University, India

Co-Chairs of Advisory Committee: Dr. Duncan Moore Hank Walker
Dr. Gwan Choi

Testing of delay defects is necessary in deep submicron (DSM) technologies. High coverage delay tests produced by automatic test pattern generation (ATPG) can be applied during wafer and package tests, but are difficult to apply during the board test, due to limited chip access. Delay testing at the board level is increasingly important to diagnose failures caused by supply noise or temperature in the board environment. An alternative to ATPG is the built-in self test (BIST). In combination with the insertion of test points, BIST is able to achieve high coverage of stuck-at and transition faults. The quality of BIST patterns on small delay defects is an open question. In this work we analyze the application of BIST to small delay defects using resistive short and open models in order to estimate the coverage and correlate the coverage to traditional delay fault models.

# ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Hank Walker, for his guidance and support throughout the course of this research and during my entire graduate studies at Texas A&M University. I'm very thankful for his help and motivation and I shall always cherish my interactions with him.

I would also like to thank Dr. Gwan Choi and Dr. Weiping Shi for agreeing to serve on my committee and for their support throughout the course of my research.

I would also like to thank my friends and the faculty and staff of the Department of Electrical and Computer Engineering for making my time at Texas A&M University a great experience.

Finally, thanks to my parents and my brother, Prithivi, for their encouragement and love throughout my studies at Texas A&M University.

# TABLE OF CONTENTS

Page

# LIST OF FIGURES

FIGURE Page

FIGURE Page

# LIST OF TABLES

# 1. INTRODUCTION

A *delay defect* is an increase in delay caused by one or more of the following factors: a spot defect that causes a resistive open or short, process variations such as variations in transistor channel length or threshold voltage, and capacitive coupling between adjacent lines of a circuit. All these factors may result in an increase in circuit delay, which will cause timing failure if the increase exceeds the circuit timing margin. Delay defects in a manufactured circuit do not affect the logic functionality of the circuit; they change only the delay of some gates, thus altering the speed at which the circuit can operate. This increase in delay is especially critical for the longest paths of the circuit, which typically have the lowest *slack* (timing margin). The *slack* is defined as the difference between the arrival time of a data signal and its required arrival time, which is the active transition (edge) of the clock (minus the setup time). A delay increase that exceeds the slack causes a *delay fault*. Due to process variations, temperature, power supply noise, and capacitive coupling, long paths may become the longest (critical) paths, so a small delay defect test must cover all of these long paths.

Delay tests are readily generated with automatic test pattern generation (ATPG) software. These patterns may target different delay fault models, such as the transition, gate or path delay fault [1]. Test data volume (and so tester memory and test application time) can be greatly reduced by using compression [2].

---

This thesis follows the style of *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.*

Even if a chip passes wafer and package testing, it must still be tested once it has been placed on a board. Boards must be tested due to the different temperature and power supply noise environment that the chips may find themselves in. This different environment may cause delay failures. Chips that fail on the board must be identified, so that they can be replaced. This requires application of a delay test. Traditionally the delay test has been performed using system functional tests. High-coverage system tests are very difficult to create and apply, and can have long execution times. The ideal situation would be to apply ATPG-generated delay test patterns to chips in the board environment.

Even with compression, the volume of data that must be sent to and received from a chip during an ATPG-based delay test is substantial. This data volume can be readily handled during wafer and package testing, where multiple high-speed test channels can be connected to the chip. Once the chip is placed on the board, it will have at most a low-speed IEEE 1149 boundary scan interface. This interface is too slow to permit economical application of ATPG-generated delay test patterns.

An alternative to ATPG pattern application during board test is built-in self test (BIST). During BIST test, on-chip hardware generates delay test patterns, and stores the result syndrome. The BIST can operate at high speed while being controlled via the slow IEEE 1149.1 channel. The central challenge is achieving high-coverage patterns with low hardware overhead and test application time.

BIST is also very useful for providers of intellectual property (IP) *cores* [3]. Equipping the cores with BIST helps to hide the intellectual property of the design, as

less test information about the core has to be given to the user. In addition, BIST-based core test simplifies the system-on-chip (SOC) design problem. While internal clock frequencies have risen by 30% per year, the accuracy of external test equipment has improved at a rate of only 12% per year [4]. BIST can track the rising internal clock frequencies, but externally applying delay tests to a core is becoming very difficult.

**1.1 Previous Work**

Defects that occur during manufacturing may cause logic circuits to malfunction at the desired clock rate and thus violate timing specifications. Such failures are modeled as delay faults. These faults may not show themselves at speeds lower than the desired clock speed. Special two-pattern test vectors (V1 and V2) are required for detecting delay faults [5]. Application of two vector patterns causes transitions to be propagated through the circuit and be captured at an observation point. These transitions are checked for excessive propagation delays beyond the functional clock period. Unfortunately, the structural limitations of scan circuits allow only two types of two-pattern delay tests to be applied in a scan design: Launch-on-Shift (LOS) and Launch-on-Capture (LOC) [6][7].

Launch-on-Capture (LOC) is also known as broad-side delay test. A broad-side delay test is a form of a scan-based delay test, where the first vector of the pair is scanned into the chain, and the second vector of the pair is the combinational circuit's response to this first vector [7]. This delay test form is called "broad-side" since the second vector of the delay test pair is provided in a broad-side fashion, namely through the logic. Launch-on-Shift (LOS) is also known as skewed-load transition test where both the first and second

vectors of the delay test pair are applied by the scan hardware [6], with the second vector being a one-bit shift of the first vector.

In [7], a number of methods of performing delay testing have been explored. In the first method, an independent set of random pattern pairs were applied to the circuit under test (CUT). The vectors were generated independently such that there was no correlation between V1 and V2. The second method was the broad-side test where V1 was a pseudorandom vector that was independently generated, and the V2 vector was the combinational circuit's response to the first vector. The third method was the skewed-load test where V2 was a one-bit-shift of its V1 predecessor. The transition fault coverage for skewed-load was considerably better than for broad-side.

In [7], a number of hybrid or extended methods were also explored. The fourth method was "Skewed + B-side" where first skewed-load patterns were applied and then broad-side patterns were applied. The fifth method applied skewed-load patterns when the latches in the scan path were reordered to enhance delay test performance. It is possible to reorder the position of the latches along the scan in order to minimize the shift dependency effect on the combinational logic. This re-ordering can substantially increase the transition fault coverage attainable by skewed-load, but at the cost of increased scan chain routing area and delay. The sixth method of delay test was using skewed-load patterns in combination with broad-side with reordered scan latches.

Though LOS tests display somewhat better coverage than LOC and generally achieve this coverage with significantly fewer test patterns, in practice, only LOC can be applied

to most circuits because LOS requires a high speed global scan enable signal, which is expensive to implement.

To address the requirement of a high speed global scan enable signal for LOS testing, a new scan flip-flop was presented in [8] that could support complete LOS testing without any need for a fast scan enable signal. Using the traditional slow scan enable and the clock signal already available in the flip-flop, the new Delay Test Scan Flip-flip (DTSFF) internally achieves a properly timed transition from the scan shift mode (at launch) to functional mode (for response capture) to support LOS tests. The new flip-flop incorporates a small amount of additional hardware.

The basic DTSFF was further enhanced using some additional logic and a second slow speed control signal to allow a choice between the LOC and LOS modes, so that both types of tests can be applied. In [9], it was shown that simple retiming of the slow scan enable signal by the tester can allow the basic DTSFF to implement both LOS and LOC tests, eliminating the need for the extra logic and second global control signal required by [8]. In [9], an efficient implementation of the DTSFF was presented that requires only a single extra six-transistor AOI(1,2) gate beyond the traditional multiplexer-D flip-flop design.

The methods presented so far enable us to perform LOC and LOS tests by which only the first vector in a delay test pair needs to be generated by a source. A number of techniques have been proposed to improve delay fault coverage using BIST by which both vectors in a delay test pair can be generated by the BIST test pattern generator (TPG). The most popular among these is single input change (SIC) TPG where the first

vector is generated by a linear feedback shift register (LFSR) and the second vector which differs from the first vector by a single bit is generated using a shift register and XOR gates. To improve fault coverage, the first vector is weighted [10] using multiple sets of weights. However, the actual weights used are determined based on an ATPG generated test set. The area overhead of the SIC TPG design was reduced by reducing the number of shift register stages in [11]. The input cones of the output lines were identified and it was found that compatible inputs could be changed simultaneously without modifying the delay fault coverage. Thus the length of the test sequence could be reduced without modifying the delay fault coverage [11]. The main disadvantage of all these methods is that since both vectors are generated by the BIST TPG, this approach requires hold scan cells [12] to preserve the first vector as the second vector is being scanned in.

A number of weighted random techniques have been presented in literature. The concept of weighted random test-pattern generation was introduced in [13] where a weighted pattern generation technique employing an LFSR and a combinational circuit was first described. The combinational circuit inserted between the output of the LFSR and the CUT is to increase the frequency of occurrence of one logic value while decreasing the other logic value. This approach may increase the probability of detecting those faults that are difficult to detect using the typical LFSR pattern generation technique. These faults are called random pattern resistant (RPR) faults. Several programmable probabilities or weight sets can be used to further increase each circuit's fault coverage [14]. However, using programmable weights requires storing the values

of the control inputs on-chip. Use of several weights also increases the area overhead due to weighted random testing.

True and inverted outputs of *k*-input AND gates [15] were used to realize weights $\{2^{-k}$, 0.5 and $1-2^{-k}\}$ for the stuck-at fault model. The input probability (the value of *k*) was decided based on an ATPG-generated deterministic test set for faults that were not detected by unweighted LFSR. A low hardware overhead scan based 3-weight weighted random BIST was proposed in [16] for the stuck-at fault model where only three weights, 0, 0.5 and 1 were assigned. Since only three weights were used, circuitry to generate weights was simple; weight 1 (0) was obtained by fixing a signal to 1 (0) and weight 0.5 by driving a signal by an output of an LFSR. ATPG was used to generate suitable test cubes from which weight sets were calculated for RPR faults. It was shown in [17] that using Markov sources as pseudo-random pattern generators in scan BIST results in tests that achieve 100% fault efficiency at much lower area overhead and reduced test lengths compared to the low area overhead method presented in [16]. These methods determine the weight sets by using ATPG generated deterministic patterns. This is essentially a mixed-mode approach. As will be described later, our approach of determining weights is not based on deterministic patterns.

# 2. BIST APPROACHES

BIST is divided into several types: BIST for memory arrays [18], data buses/networks and I/Os [19], and logic [20]. High-coverage tests for memories and data transfer circuits can be implemented in small state machines. In contrast, logic BIST (LBIST) typically cannot achieve high test coverage without significant design modification. There has been little work addressing small delay defects in logic. Therefore the rest of this paper focuses on LBIST.

LBIST consists of a test pattern generator (TPG) that applies tests to the circuit under test (CUT), and a test response analyzer (TRA) that compactly stores test responses for analysis. The TPG is most commonly a pseudo-random pattern generator (PRPG), usually implemented as a linear feedback shift register (LFSR). An example of an LFSR is shown in Figure 1. An *n*-bit LFSR implementing a primitive polynomial will have a sequence of $2^n$-1 before repeating.

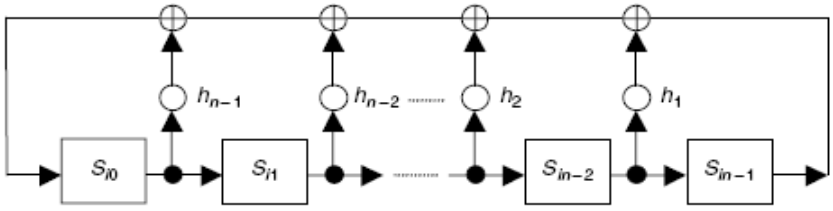


Figure 1. An *n*-stage LFSR [21]

The test patterns can be applied to the circuit directly on its inputs, or they can be scanned in via scan chains. Similarly, the output response can be taken directly from the circuit outputs, or scanned out via scan chains.

### 2.1 Test Patterns Directly Applied to CUT

The test patterns can be directly applied to the circuit by having an *n*-bit LFSR feed *n* circuit inputs. The circuit inputs are made up of both the primary inputs and the pseudo-primary inputs (scan cell outputs). These circuits also form a Multiple Input Signature Register (MISR). A MISR is an LFSR with an additional XOR gate on each flip-flop input that superimposes the inputs on the LFSR value. At the end of testing, the value remaining in the MISR (the signature) is scanned out and compared to the good value. If the LFSR has a long sequence, then there is a very high probability than an error propagated to the MISR inputs will cause the MISR signature to be different than the correct value.

The Built-In Logic Block Observer (BILBO) technique [22] combines the LFSR and MISR into one circuit. The advantage of BILBO is that a test pattern can be applied on each clock cycle. At full functional clock speed, a large number of tests can be applied in a short period of time. One of the disadvantages of this approach is that all of the inputs and scan cells in the circuit must be routed together into an LFSR. A second disadvantage is that each scan cell must have both LFSR and MISR functionality, due to loops in the logic circuit. This adds area and delay overhead to the scan cells.

### 2.2 STUMPS - Test Per Scan (TPS)

Self-Testing using MISR and parallel SRSG (STUMPS) [23] contains a parallel Shift Register Sequence Generator (SRSG) and a MISR, as shown in Figure 2 [1]. The Parallel SRSG (PRPG) is normally realized using an LFSR. The length of the LFSR is equal to the number of primary inputs and scan chain inputs, which is much smaller than

the number of pseudo-primary inputs. The LFSR, MISR and scan chain shift are all clocked at the same rate. The STUMPS architecture uses the same scan chains that are implemented for standard scan testing. The area overhead of the STUMPS approach is small, since there is only one LFSR bit at the input and one MISR bit at the output of each scan chain.

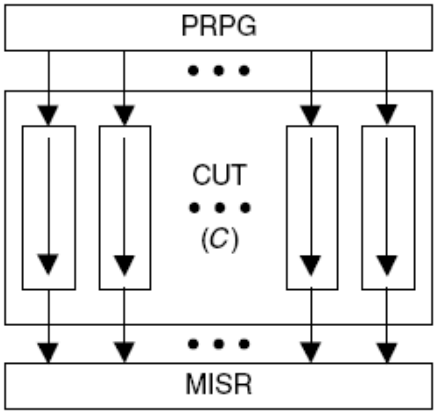


Figure 2. STUMPS configuration [1]

In the STUMPS - Test Per Scan (STUMPS-TPS) approach, the scan chains are shifted for their entire length, feeding in test data from the LFSR. Then the scan enable signal is turned off, a system clock is applied to capture the CUT outputs of the test pattern, and then the scan enable turned on, and the test results scanned into the MISR (while the next test pattern is scanned in).

A delay test requires transitions on the CUT inputs. There are three approaches that can be used. In the enhanced scan (ES) approach, the first test pattern is stored in latches on the outputs of the scan cells. Then the second test pattern is scanned in. When the

latches capture this second test pattern, transitions are generated where the first and second pattern differ. The area and delay overhead of the latches makes this approach infeasible. The second pattern can be generated by a one-bit shift of the first pattern, in an approach known as launch-on-shift (LOS). The advantage is that this is very simple. The disadvantage is that a high-speed scan enable signal must be distributed to all scan cells. In the launch-on-capture (LOC) approach, the test pattern is scanned in, then the scan enable is turned off, and the system clock is applied twice at full speed. The first clock causes the scan cells to capture the CUT response to the first pattern, generating transitions, and the second clock captures the delay test response. LOC has the advantage that it does not require a high-speed scan enable, and it operates the circuit in a functional mode. It has the disadvantage that more test patterns are required to achieve a given fault coverage. In LOC within STUMPS [24], the CUT only has to be modified at the input and output of the scan chains.

**2.3 STUMPS - Test Per Clock (TPC)**

STUMPS - Test Per Clock (STUMPS-TPC) is a BIST scheme intermediate between BILBO and STUMPS-TPS. In this approach the scan chains shift only one bit between capture cycles. This also provides a form of LOS test. Because only one bit at a time is shifted out each scan chain before the result is overwritten, the scan chains must be configured as MISRs, rather than a MISR on the scan chain outputs. The advantage of STUMPS-TPC is that test patterns can be applied one per clock cycle, as in BILBO, and there is less area overhead since the LFSR is external to the scan chains. The disadvantage is the need to implement the large MISRs.

## 2.4 Weighted Random Patterns and Test Points

The basic LFSR provides pseudo-random test patterns that can detect many faults. However, faults whose detection requires many specific values in the circuit are unlikely to be detected. These are termed *random pattern resistant* (RPR) faults. Delay defects on the longest paths in the circuit are RPR, since the longer the path, the more necessary assignments to propagate along it.

Several approaches have been used to improve the coverage of RPR faults. The first approach is to reseed the LFSR. Essentially, the LFSR skips over a long sequence of states where it detects few or no faults [25]. Prior research on path delay faults [26] has shown that a test pattern that detects one hard-to-detect fault is very unlikely to detect another fault. As a result, the test process very quickly approaches the need to apply a new LFSR seed for each fault.

A second approach to detecting RPR faults is to insert *test points*. Test points are extra control or observation points in the circuit that increase the probability of detecting the RPR faults. Test point insertion algorithms balance improvement in fault coverage against area, power and delay increase [1]. We do not consider test points in this research since we are focused on coverage of small delay defects on critical paths. Because the paths are critical, test points cannot be inserted on them.

The third approach to improving RPR coverage is to use weighted random pattern (WRP) generation (WRPG). There are many WRP algorithms in the literature [10][13][14][15][16]. The LFSR is modified so that the distribution of the test patterns is not random, and are targeted at the undetected faults. Prior research on WRPG has

concentrated on testing stuck-at and transition faults, often in conjunction with test point insertion. True and inverted outputs of *k*-input AND gates [15] were used to realize weights $\{2^{-k}$, 0.5 and $1-2^{-k}\}$. The input probability (the value of *k*) was decided based on an ATPG-generated deterministic test set for faults that were not detected by ordinary LFSR.

Prior research has been done on applying single input change (SIC) test patterns to transition faults. The single transition is produced by having all LFSR outputs feed XOR gates, and the side input of one XOR gate at a time is changed. To improve fault coverage, the first vector can be weighted [10] using several different weights. The weights used are determined based on an ATPG generated test set. The area overhead of the SIC TPG design was reduced by reducing the number of shift register stages in [11]. The input cones of the output lines were identified and it was found that compatible inputs could be changed simultaneously without modifying the delay fault coverage. Thus the length of the test sequence could be reduced without modifying the delay fault coverage [11]. The main disadvantage of all these methods is that since both vectors are generated by the BIST TPG, this approach requires hold scan cells [12] in a form of enhanced scan to preserve the first vector as the second vector is being scanned in.

Transition fault test results based on LOC have been reported in [7][9]. In this research we will focus on weighting the outputs of the LFSR to improve LOC delay fault coverage.

# 3. WEIGHTED RANDOM PATTERN GENERATION

Weighted random pattern generation biases the probability of an LFSR output being a 0 or 1. These biases were implemented using two-, three- and four-input AND and NAND combinational gates separately to weight the output bits of the LFSR prior to shifting into the scan chain. The AND gates increase the probability of a 0 while NAND gates increase the probability of a 1. The hardware overhead is low since a single AND gate and an XOR gate are sufficient to realize either an AND or a NAND weight. Apart from these combinational gates, OA21 and OAI21 gates were also used to obtain other weights. The different weights considered in this research are listed in Tables 1 and 2. More combinations and higher and lower weight values did not provide any benefit.

Table 1. Non-inverting gate weights

| NON-INVERTING COMBINATIONAL GATES | | | | | |
|---|---|---|---|---|---|
| | PR | and2 | and3 | and4 | oa21 |
| 0 | 0.5 | 0.75 | 0.875 | 0.9375 | 0.625 |
| 1 | 0.5 | 0.25 | 0.125 | 0.0625 | 0.375 |

Table 2. Inverting gate weights

| INVERTING COMBINATIONAL GATES | | | | | |
|---|---|---|---|---|---|
| | PR | nand2 | nand3 | nand4 | oai21 |
| 0 | 0.5 | 0.25 | 0.125 | 0.0625 | 0.375 |
| 1 | 0.5 | 0.75 | 0.875 | 0.9375 | 0.625 |

The probability of an output bit of an LFSR being 0 or 1 is 0.5. However, for hard to sensitize scenarios like setting the output of a 10-input AND gate to logic 1 requires setting all the 10 inputs to 1. With an ordinary LFSR, the probability that the output is set to logic 1 is $(0.5)^{10}$ which is less than 1/1000. However, if we use a 4-input NAND gate to weight the outputs of an LFSR, there is a 0.9375 probability of generating logic 1 at the output of the LFSR. For the 10-input AND gate, with the 4-input NAND weight the probability that the output is set to logic 1 is $(0.9375)^{10}$ which is just more than 1/2. The probability of getting a logic 1 at the output of the 10-input AND gate increases by more than 500 times by using a 4-input NAND weight over an ordinary LFSR. Hence, to obtain an output probability close to 1, a multi-input NAND gate is to be used and to obtain an output probability close to 0, a multi-input AND gate is to be used.

For our experiments, a single scan chain is constructed by stitching together all the scan cells in the design. A single gate is sufficient at the output of the LFSR to weight the single bit that is being fed into the scan chain. Hence, if a 4-input AND gate is used to weight the bit that is fed into the scan chain, out of every 16 bits fed into the scan chain, there is a probability that 15 bits are of logic 0 and 1 bit is of logic 1. Hence, the result is that too many zeros are likely to be fed into the scan chain. When too many zeros are likely to get fed into the scan chain, the probability of sensitizing paths with gates such as AND and NAND which require the side-inputs to have a logic of 1 is low. These gates have logic 0 as controlling values and hence filling the scan chain with zeros results in blocking all these paths and hence the path delay fault coverage goes down. Similarly using a 4-input NAND gate to weight the bit that is being fed into the scan

chain results in the scan chain being filled with too many ones. This results in blocking paths composed of OR and NOR gates which have a controlling value of logic 1. The 4-input AND and NAND gates bias the probability close to 0 and 1 respectively, thus providing a large bias from the usual probability of 0.5.

Thus, to avoid blocking of paths, gates which provide a small bias from 0.5 such as 3-input and 2-input AND and NAND gates have been used to improve the fault coverage. This tells us intuitively that gates such as OA21 and OAI21 which provide an even smaller bias from 0.5 are likely to give better fault coverage than two- and three-input AND and NAND weights. Thus, OA21 and OAI21 weights have also been used in our experiments to weight the output bit of the LFSR.

The usual method used for realizing a weight (probability) of $1\text{-}2^{-k}$ is to invert the output of a *k*-input AND gate (a *k*-input NAND gate) [14][15]. For example, a 2-input NAND gate realizes a 0.25 probability for logic 0 and a 0.75 probability for logic 1. However, a probability of $1\text{-}2^{-k}$ can also be realized using a *k*-input OR gate. For example, a 2-input OR gate also realizes a 0.25 probability for logic 0 and a 0.75 probability for logic 1. These two gates behave differently when used for improving fault coverage despite realizing the same probability. The reason for this can be determined by observing the truth table of 2-input NAND and OR gates which is given in Table 3.

Table 3. Truth table for two input NAND and OR gates

| IN A | IN B | NAND OUT | OR OUT |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |

For an input of 00, the NAND gate has an output of logic 1 whereas the OR gate has an output of logic 0. Similarly, for input 11, the NAND gate has an output of 0 whereas the OR gate has an output of 1. This results in different bits being fed into the scan chain when the NAND and OR gates have inputs of 00 and 11. The different bits fed into the scan chain result in detecting different faults. By the same reasoning a *k*-input NOR gate detects different faults when compared with a *k*-input AND gate in realizing a probability of $2^{-k}$ and ends up providing different fault coverage.

### 3.1 Fault Models

This research considers both transition faults (TFs) and resistive opens and shorts. BIST for transition faults has been extensively studied. In this research, the TF coverage forms an upper bound on the delay fault coverage that can be achieved for a given BIST design. We are also interested in whether the TF coverage can be used as an estimate of the resistive open and short coverage.

*3.1.1 Transition Faults*

The transition fault model assumes that a gate input or output has a slow-to-rise or slow-to-fall fault [27]. These delays are assumed to be so large that any path through these fault sites is slow. To detect these delay faults, the corresponding slow transition must be generated and then propagated to an observation point. In the context of BIST, sensitizing most transition faults is easy, since most lines have a significant chance of toggling. The challenge is that a propagation path may need many necessary assignments, which makes the probability of sensitization low.

*3.1.2 Resistive Open and Short Faults*

The transition fault model is adequate for detecting gross delay defects. But it cannot detect small delay defects caused by resistive shorts or opens. Detecting these faults requires both propagating along the longest paths through them, but sensitizing the fault condition (e.g. setting the bridged node to the opposite value). If BIST cannot sensitize the longest paths in the circuit, it may have high transition fault coverage, but poor resistive short and open coverage. The longer the path sensitized, the smaller the open resistance detected. Similarly, the longer the path tested, the larger the short resistance detected. Detecting small open resistances and large short resistances is increasingly important, since circuit optimization makes the design increasingly susceptible to small delay defects (SDDs).

**3.2 Experimental Methodology**

We evaluated a number of different BIST approaches on ten ISCAS89 benchmark circuits [28], using both the transition fault model and the resistive short and open

model. The *CodSim* fault simulator [29] was used to evaluate all three fault models. In this research, we assumed that there was no MISR fault masking, so the MISR was not considered in the analysis. Given the timing realities of modern circuits, LOC was used to generate the second test pattern in the tests.

The BIST methods considered were BILBO, STUMPS-TPS and STUMPS-TPC, along with different weighting methods. In the STUMPS techniques, a single scan chain was used, fed by the last bit of the LFSR.

Primitive polynomials have been used to realize the LFSR in the BILBO method. A type II 32-bit LFSR was used for STUMPS-TPS and STUMPS-TPC methods. For the weighting schemes, the last few bits of the LFSR were fed as inputs to the combinational gate realizing the weight. For example, for a 4-input NAND weight, the last four bits of the LFSR were given as inputs and the output of the NAND gate was fed into the single scan chain.

# 4. EXPERIMENTAL RESULTS AND ANALYSIS

## 4.1 Comparison of Different BIST Approaches

The three different BIST approaches were applied using the transition fault (TF) model. For each method, 10,000 test patterns were applied. For the BILBO method, this corresponds to 10,000 clock cycles. For STUMPS-TPC, this corresponds to 10,000 one-bit scan chain shifts. For STUMPS-TPS, this corresponds to 10,000 full scan shifts, which is slower by a factor of the scan chain length. Intuition suggests that BILBO and STUMPS-TPS should have comparable coverage, while STUMPS-TPC should be lower, due to the greater correlation between test patterns. However, the results in Table 4 show that all three methods have similar coverage.

Table 4. Transition fault coverage

| ISCAS89 Circuit | Fault Count | BILBO FC (%) | STUMPS-TPC FC (%) | STUMPS-TPS FC (%) |
|---|---|---|---|---|
| s1423 | 2846 | 71.43 | 72.13 | 71.50 |
| s1488 | 2976 | 79.83 | 79.70 | 79.83 |
| s1494 | 2988 | 79.61 | 79.48 | 79.61 |
| s5378 | 10590 | 71.10 | 72.59 | 72.45 |
| s9234 | 18468 | 57.59 | 58.92 | 60.08 |
| s13207 | 26358 | 67.38 | 67.23 | 68.88 |
| s15850 | 31694 | 55.86 | 56.35 | 55.89 |
| s35932 | 71224 | 71.54 | 71.54 | 71.54 |
| s38417 | 76678 | 82.07 | 86.88 | 87.26 |
| s38584 | 76864 | 61.07 | 65.05 | 64.94 |

The BILBO and STUMPS-TPC methods provide similar fault coverage (FC) to STUMPS-TPS with a much lower application time, but a much higher area, and potential impact on circuit delay. Given that the impact on the design is the most critical factor, it can be concluded that the STUMPS-TPS approach is the best one to use. In the following, we only consider STUMPS-TPS.

**4.2 Evaluation of WRPG**

The different weights were evaluated for their impact on fault coverage. The unweighted STUMPS-TPS LFSR TF results are shown in Table 5 for reference. As in Table 4, the results are for 10,000 test patterns.

Table 5. Unweighted LFSR coverage

| ISCAS89 Circuit | Total Faults | Faults Detected | Fault Coverage (%) |
|---|---|---|---|
| s1423 | 2846 | 2035 | 71.5 |
| s1488 | 2976 | 2376 | 79.83 |
| s1494 | 2988 | 2379 | 79.61 |
| s5378 | 10590 | 7673 | 72.45 |
| s9234 | 18468 | 11096 | 60.08 |
| s13207 | 26358 | 18156 | 68.88 |
| s15850 | 31694 | 17715 | 55.89 |
| s35932 | 71224 | 50958 | 71.54 |
| s38417 | 76678 | 66913 | 87.26 |
| s38584 | 76864 | 49917 | 64.94 |

The results for transition fault testing ISCAS89 circuits with LOC have already been reported elsewhere [7][9]. Table 6 compares our fault efficiency (STUMPS-TPS FE column) and fault coverage (STUMPS-TPS FC column) results along with their results (columns labeled Savir [7] and Gefu [9]). Our fault efficiency results are comparable to those in [7] and [9]. The slight difference in results can be attributed to different simulation environments. They used a simulation-based ATPG program to generate transition delay fault test patterns whereas we use a 32-bit type II LFSR based on a STUMPS-TPS approach to generate our results. They have used 100,000 pseudo-random test vectors whereas our results are for 10,000 pseudo-random test vectors. We report only fault coverage results in the following sections for simplicity.

Table 6. Comparison of LOC results

| ISCAS89 Circuit | Savir FE [7] 100k | Gefu FE [9] 100k | STUMPS-TPS FE 10k | STUMPS-TPS FC 10k |
|---|---|---|---|---|
| s1423 | 88.65 | 87.1 | 82.89 | 71.5 |
| s1488 | 91.3 | 87.4 | 87.18 | 79.83 |
| s1494 | 90.68 | 86.98 | 87.39 | 79.61 |
| s5378 | 95.82 | 89.61 | 78.55 | 72.45 |
| s9234 | 72.55 | 74.71 | 70.37 | 60.08 |
| s13207 | 82.86 | 82.38 | 77.22 | 68.88 |
| s15850 | 65.14 | 78.82 | 66.38 | 55.89 |
| s35932 | 85.98 | NR | 82.16 | 71.54 |
| s38417 | NR | NR | 88.77 | 87.26 |
| s38594 | NR | NR | 71.51 | 64.94 |

NR implies that the results were not reported

*4.2.1 The Best Single Weight*

An experiment was carried out to find the weight which gives the highest TF delay fault coverage under the LOC approach. It is possible to design BIST circuits with multiple weights and multiplex among them, but for low BIST area overhead, one weight is desirable. The weights in Tables 1 and 2 were applied separately to the ISCAS89 circuits for 10,000 patterns. The weight was computed using the last bits of the LFSR. The *additional* faults detected beyond the 10,000 original LFSR patterns are shown in Table 7.

Table 7. Additional faults detected by weights

| Circuit | and2 | nand2 | and3 | nand3 | and4 | nand4 | oa21 | oai21 |
|---|---|---|---|---|---|---|---|---|
| s1423 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| s1488 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| s1494 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| s5378 | 41 | 134 | 4 | 81 | 0 | 76 | 0 | 76 |
| s9234 | 83 | 38 | 173 | 30 | 100 | 14 | 100 | 14 |
| s13207 | 1141 | 481 | 781 | 442 | 456 | 347 | 456 | 347 |
| s15850 | 525 | 577 | 460 | 263 | 267 | 131 | 267 | 131 |
| s35932 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| s38417 | 1355 | 1984 | 758 | 1791 | 425 | 1315 | 425 | 1315 |
| s38594 | 1102 | 808 | 717 | 737 | 280 | 554 | 280 | 554 |

The single best weight for each circuit is shown in Table 8 along with the additional faults detected by it. For comparison, the number of additional faults detected by applying 10,000 unweighted patterns is shown.

Table 8. Best weight for each circuit

| ISCAS89 Circuit | Max. additional single weight faults 10k | Best weight | Additional LFSR faults 10k |
|---|---|---|---|
| s1423 | 0 | none | 32 |
| s1488 | 0 | none | 0 |
| s1494 | 0 | none | 0 |
| s5378 | 134 | nand2 | 199 |
| s9234 | 173 | and3 | 464 |
| s13207 | 1141 | and2 | 584 |
| s15850 | 577 | nand2 | 454 |
| s35932 | 0 | none | 0 |
| s38417 | 1984 | nand2 | 1377 |
| s38584 | 1102 | and2 | 1205 |

As can be seen in Table 8, the additional WRPG test patterns detect slightly more TFs than the unweighted patterns. This is especially true for large circuits. For example, for circuit s38417, 10,000 additional LFSR-generated patterns detect only 1377 additional

faults whereas 10,000 additional WRPG-generated patterns detect 1984 more faults. The particular weight that achieves the highest improvement in fault coverage for every circuit is also listed in Table 8. For most circuits, the AND2 or NAND2 weight achieved the highest fault coverage. This shows that for a constraint of choosing a single weight, two-input weights are more effective in improving fault coverage than more biased weights.

### *4.2.2 Use of All Weights*

With multiplexing, several weights can be used, with different numbers of test patterns per weight. After 10,000 unweighted patterns, 10,000 additional patterns were applied for each weight, and the number of additional TFs detected recorded. The number of additional faults detected by each weight is shown in Table 9. The weights were applied from left to right, and only the new faults detected by that weight given.

Table 9. Additional faults detected by each weight applied sequentially

| Circuit | and2 | nand2 | and3 | nand3 | and4 | nand4 | oa21 | oai21 |
|---|---|---|---|---|---|---|---|---|
| s1423 | 27 | 11 | 0 | 0 | 0 | 0 | 9 | 6 |
| s1488 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| s1494 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| s5378 | 34 | 129 | 1 | 0 | 0 | 0 | 26 | 52 |
| s9234 | 249 | 583 | 48 | 39 | 7 | 0 | 108 | 55 |
| s13207 | 1135 | 437 | 92 | 31 | 2 | 0 | 104 | 29 |
| s15850 | 431 | 324 | 41 | 107 | 21 | 27 | 191 | 139 |
| s35932 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| s38417 | 1265 | 1483 | 62 | 222 | 1 | 29 | 214 | 199 |
| s38584 | 1014 | 831 | 152 | 165 | 51 | 48 | 305 | 205 |

Given that eight weights are applied, along with the unweighted patterns, a total of 90,000 patterns is applied to each circuit. For example, for circuit s5378, 34 additional faults were detected by applying the AND2 weight, then 129 new faults were detected by the NAND2 weight, and so on. Note that these faults are in addition to the 7673 faults that were detected using the 10,000 test patterns generated by the unmodified LFSR. The fault coverage obtained when applying all weights is given in Table 10.

Table 10. WRPG fault coverage

| ISCAS89 Circuit | Fault count | Initial LFSR faults | WRPG additional faults | WRPG total faults | WRPG FC (%) |
|---|---|---|---|---|---|
| s1423 | 2846 | 2035 | 53 | 2088 | 73.36 |
| s1488 | 2976 | 2376 | 0 | 2376 | 79.83 |
| s1494 | 2988 | 2379 | 0 | 2379 | 79.61 |
| s5378 | 10590 | 7673 | 242 | 7915 | 74.74 |
| s9234 | 18468 | 11096 | 1089 | 12185 | 65.97 |
| s13207 | 26358 | 18156 | 1830 | 19986 | 75.82 |
| s15850 | 31694 | 17715 | 1281 | 18996 | 59.93 |
| s35932 | 71224 | 50958 | 0 | 50958 | 71.54 |
| s38417 | 76678 | 66913 | 3475 | 70388 | 91.79 |
| s38584 | 76864 | 49917 | 2771 | 52688 | 68.54 |

From Table 10, it is clear that WRPG significantly improved the TF coverage under LOC test application. For example, the fault coverage for s38584 rose from 64.94% to 68.54%. Circuit s35932 is a special case, where WRPG did not detect any additional faults. This is because it contains a set of faults that are relatively easy to detect (and were detected by the LFSR patterns), and a set of faults that are very hard to detect (and were not detected with the WRPG).

The number of faults detected as the number of test vectors is increased is given in Figures 3-12 for ISCAS89 benchmark circuits. The test vectors axis has been scaled down by a factor of 100. A value of 600 for the test vectors in the figures corresponds to a test vector count of 60,000. The first 10,000 vectors were generated using ordinary LFSR. The remaining 80,000 vectors were generated with 8 different weights using 10,000 vectors for each weight. These figures show that the fault coverage quickly increases with applied test vector count and then plateaus. After that, the RPR faults were detected by applying weighted random patterns.

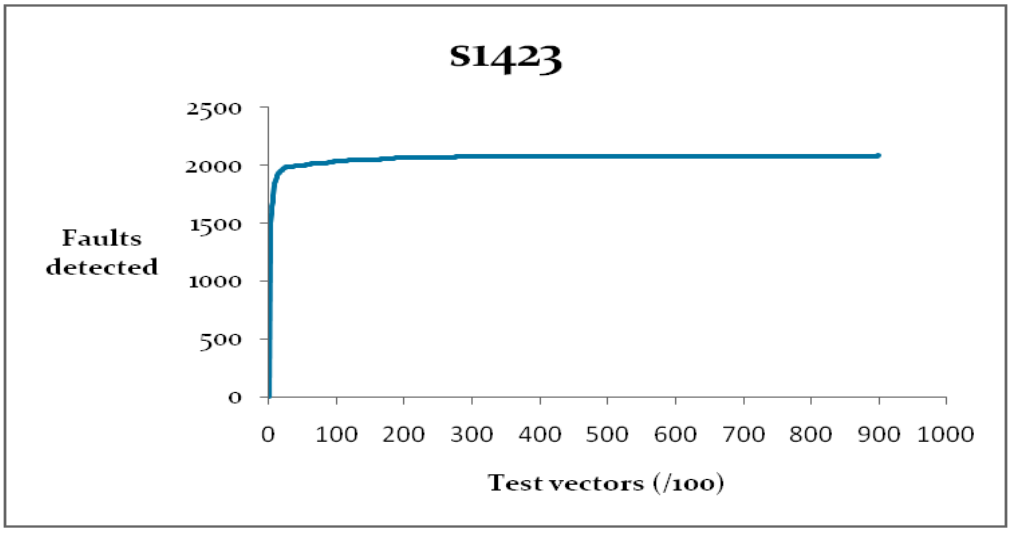


Figure 3. s1423 faults detected vs. test vectors

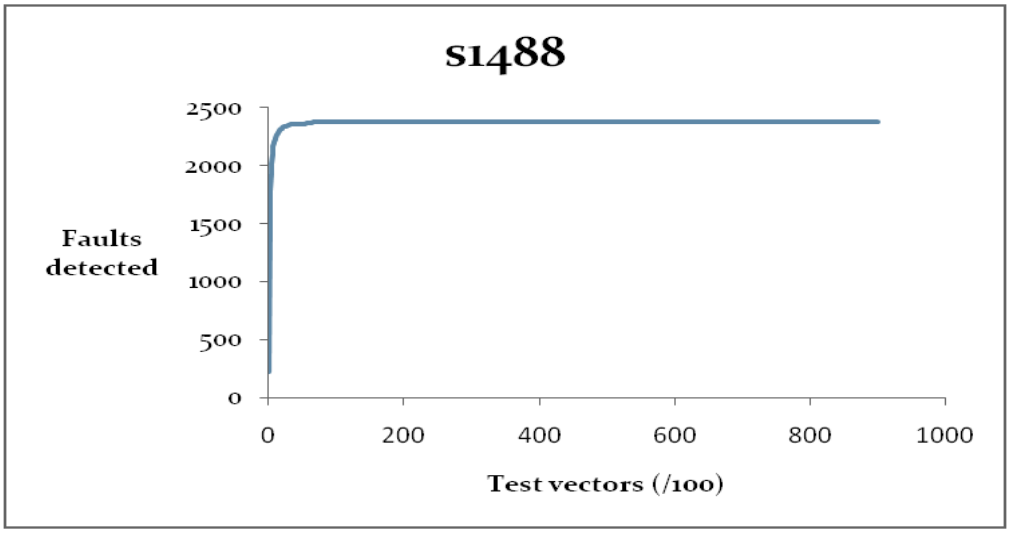


Figure 4. s1488 faults detected vs. test vectors

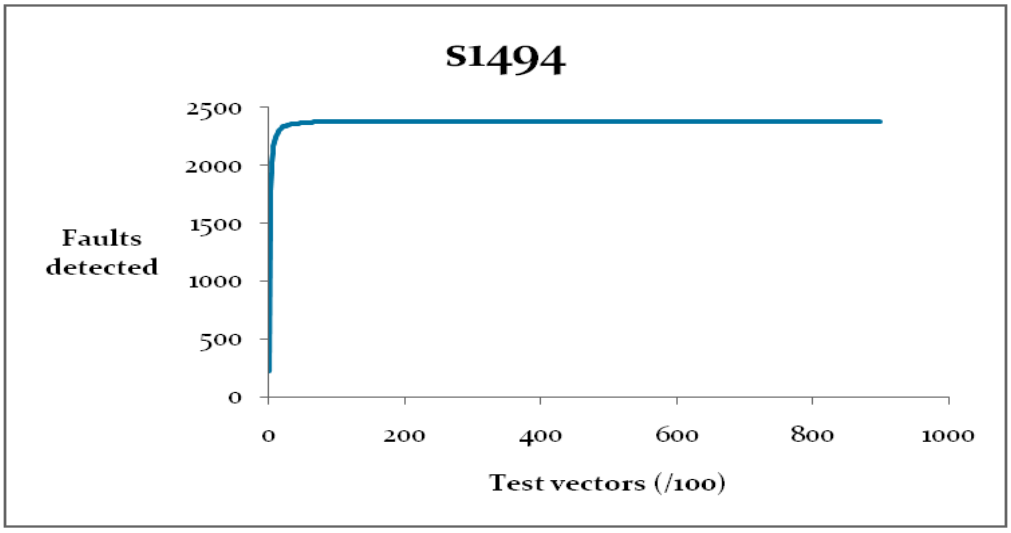


Figure 5. s1494 faults detected vs. test vectors

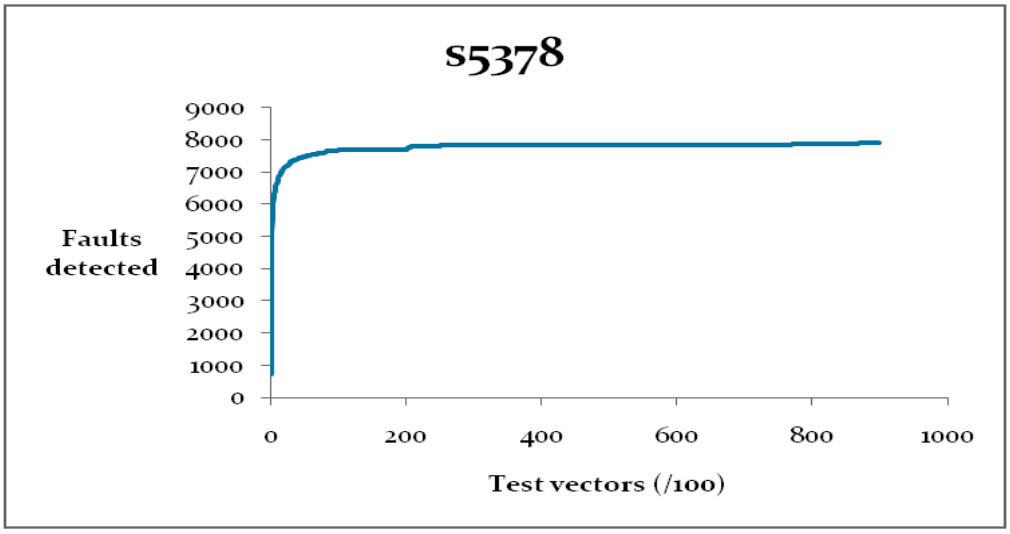


Figure 6. s5378 faults detected vs. test vectors

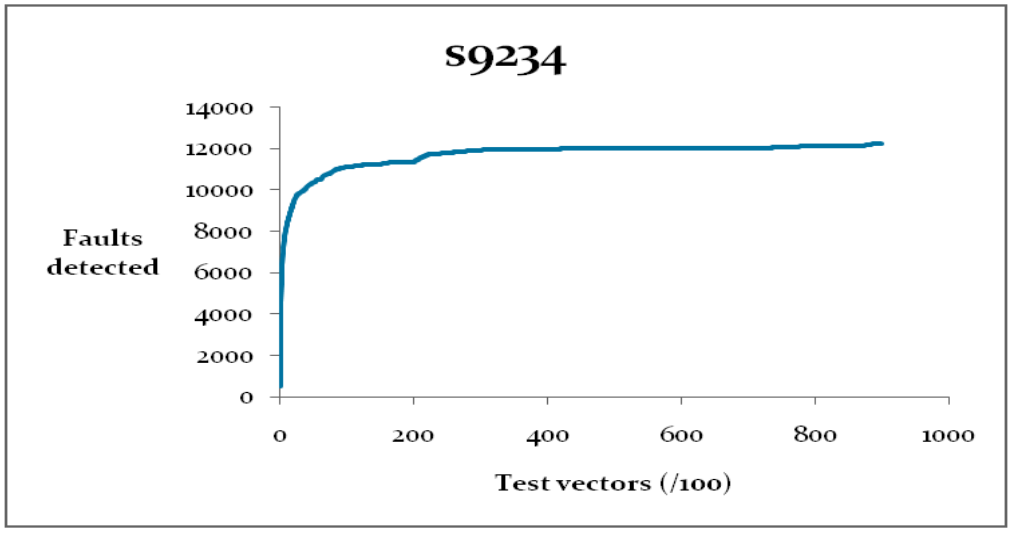


Figure 7. s9234 faults detected vs. test vectors

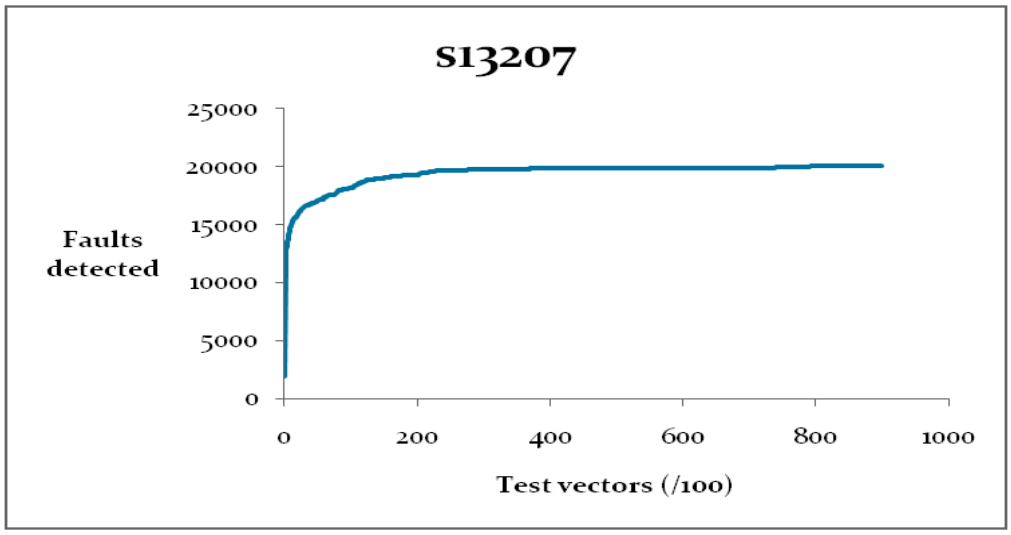


Figure 8. s13207 faults detected vs. test vectors

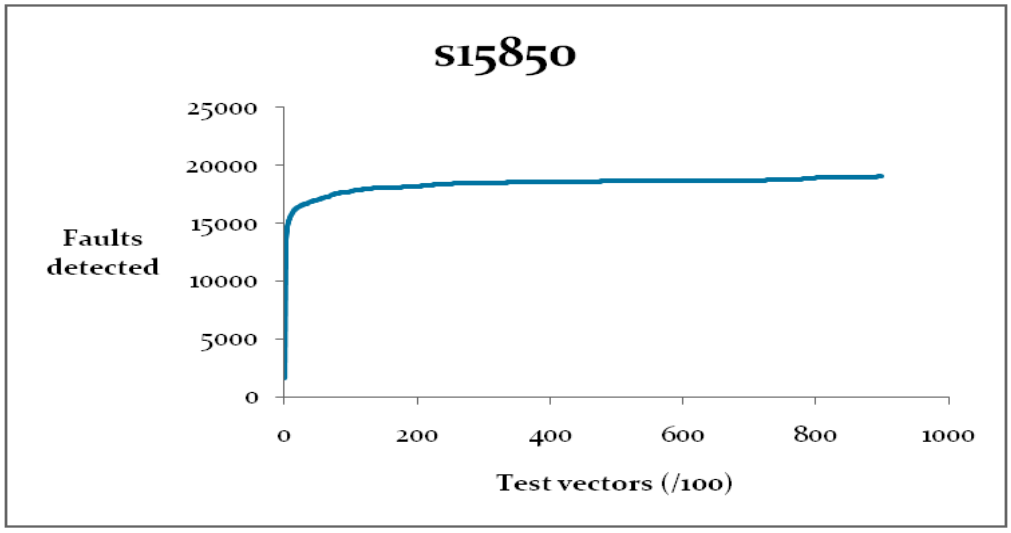


Figure 9. s15850 faults detected vs. test vectors

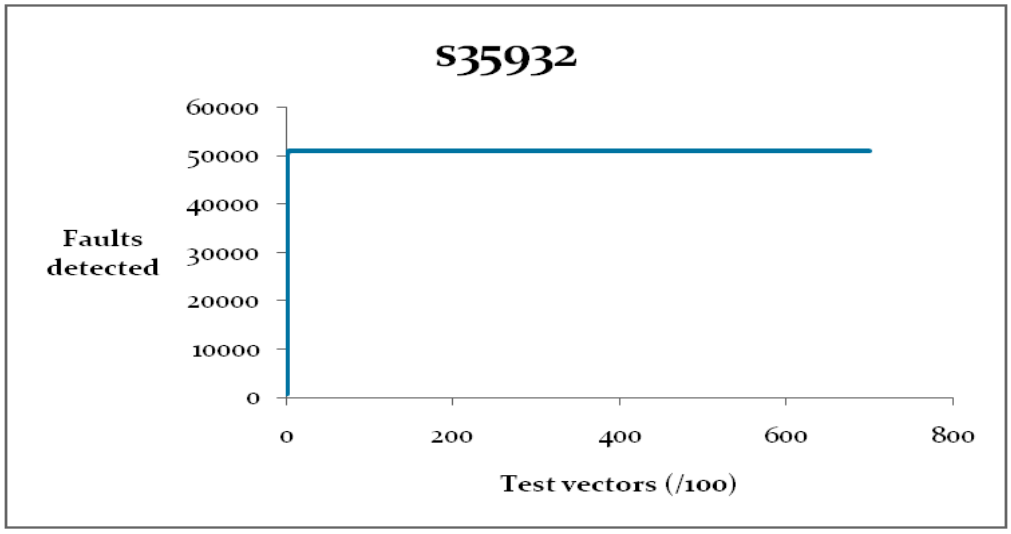


Figure 10. s35932 faults detected vs. test vectors

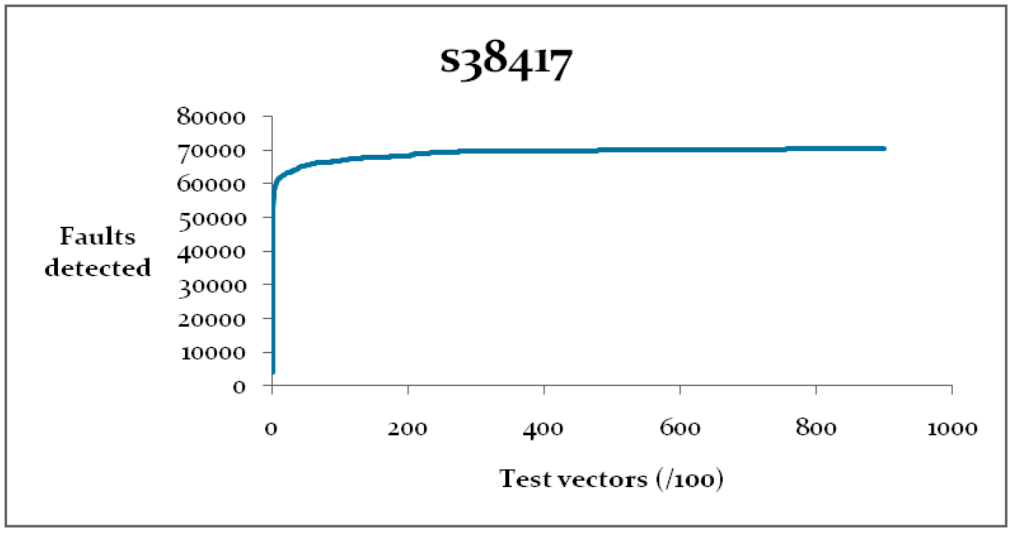


Figure 11. s38417 faults detected vs. test vectors

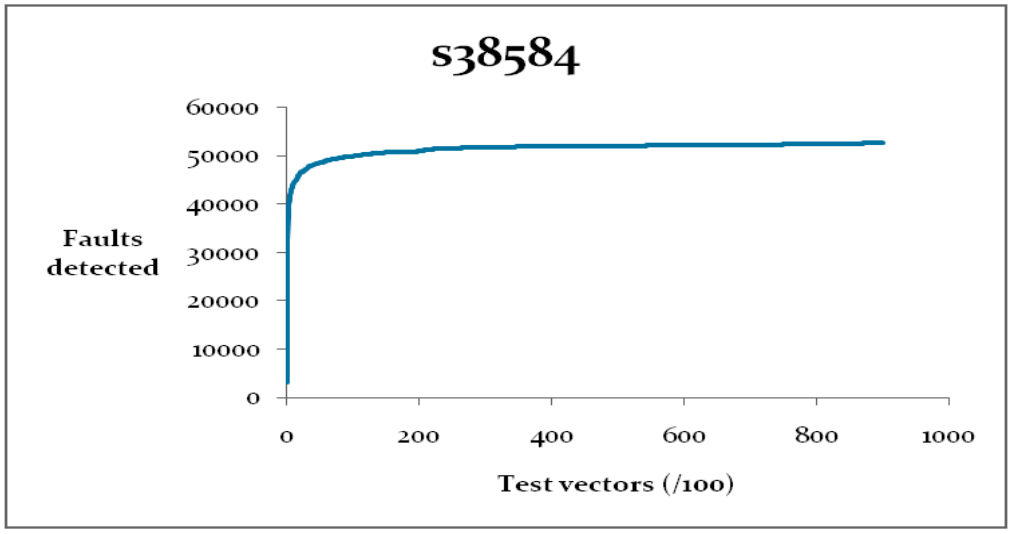


Figure 12. s38584 faults detected vs. test vectors

*4.2.3 Dependence of Weight (Probability) on Gate Type*

Experiments were conducted to determine the effectiveness of improving fault coverage when a 2-input OR gate was used over a 2-input NAND gate in realizing a

probability of 0.75 for logic 1. For the ISCAS89 circuit configurations used in these experiments, the 2-input OR gate weight detects more faults than the 2-input NAND gate. This is seen in Table 11, which lists the number of additional TF faults detected by each weight, applied separately with 10,000 patterns. Similarly, a 2-input NOR gate also results in higher fault coverage than a 2-input AND gate in realizing a probability of 0.25 (for logic 1). These experiments were conducted with a starting seed of 1111….1111 in the LFSR. These experiments clearly prove the dependence of weight (probability) realized on the type of gate used for realizing the weight, for the given starting seed.

Table 11. Fault detection for different gate types

| ISCAS89 Circuit | Total faults | Faults detected | or2 | nor2 | nand2 | and2 |
|---|---|---|---|---|---|---|
| s1423 | 2846 | 2035 | 16 | 13 | 11 | 27 |
| s1488 | 2976 | 2376 | 0 | 0 | 0 | 0 |
| s1494 | 2988 | 2379 | 0 | 0 | 0 | 0 |
| s5378 | 10590 | 7673 | 142 | 50 | 129 | 34 |
| s9234 | 18468 | 11096 | 582 | 178 | 583 | 249 |
| s13207 | 26358 | 18156 | 471 | 1147 | 437 | 1135 |
| s15850 | 31694 | 17715 | 315 | 562 | 324 | 431 |
| s35932 | 71224 | 50958 | 0 | 0 | 0 | 0 |
| s38417 | 76678 | 66913 | 1958 | 1307 | 1483 | 1265 |
| s38584 | 76864 | 49917 | 866 | 1104 | 831 | 1014 |

*4.2.4 Impact of Care Bit Density on Effectiveness of Weighting BIST Test Patterns*

*CodGen* [26] was used to generate ATPG patterns under a TF model for the ISCAS89 circuits. The average number of care bits per fault was determined for each benchmark circuit and reported in Table 12. The number of care bits gives an indication of how effective a given number of unweighted or weighted random patterns will be in detecting faults for a given benchmark circuit. Circuits s1423, s1488 and s1494 have 24.80%, 54.58% and 54.54% average care bit density per fault. These are very high care bit densities. The chance that a random pattern will set all of the care bits to their necessary values to detect a fault is low. Similarly, weighted random patterns will not help much, as can be seen in Tables 7 and 9. The other circuits (except s35932) have less than 10% average care bit density. Since the average number of care bits is relatively low, random and weighted random patterns will have more impact on the coverage. Circuit s35932 is an exception because it has on average only 6 care bits among 1763 input bits. The unweighted patterns will detect these average faults. However, these 6 care bits can be distributed anywhere within the 1763 input bits and therefore generating a test vector with bits which match the exact location of these care bits is very difficult. Therefore, these faults cannot be detected by unweighted or weighted patterns. Hence, weighting the random patterns does not provide any improvement in fault coverage for s35932.

These interpretations are based on only the average care bit percentages. The actual number of care bits can vary significantly per pattern. This analysis only provides an insight into the likely effectiveness of weighted random patterns for a given circuit.

Table 12. Effectiveness of WRPG based on average number of care bits in ATPG test vector set

| Circuit | Total TFs | No. input bits | No. ATPG vectors | Total no. care bits | Total no. Bits | care bit density | % care bit density | avg no. care bits per vector | rounded avg no. care bits per vector | STUMPS-TPS FC 10k | WRPG FC 90k |
|---|---|---|---|---|---|---|---|---|---|---|---|
| s1423 | 2846 | 91 | 1100 | 24827 | 100100 | 0.2480 | 24.80 | 22.57 | 23 | 71.5 | 73.36 |
| s1488 | 2976 | 14 | 1548 | 11829 | 21672 | 0.5458 | 54.58 | 7.64 | 8 | 79.83 | 79.83 |
| s1494 | 2988 | 14 | 1555 | 11874 | 21770 | 0.5454 | 54.54 | 7.63 | 8 | 79.61 | 79.61 |
| s5378 | 10590 | 214 | 3805 | 52342 | 814270 | 0.0642 | 6.42 | 13.75 | 14 | 72.45 | 74.74 |
| s9234 | 18468 | 247 | 4864 | 107255 | 1201408 | 0.0892 | 8.92 | 22.05 | 23 | 60.08 | 65.97 |
| s13207 | 26358 | 700 | 7552 | 92963 | 5286400 | 0.0175 | 1.75 | 12.30 | 13 | 68.88 | 75.82 |
| s15850 | 31694 | 611 | 8333 | 138542 | 5091463 | 0.0272 | 2.72 | 16.62 | 17 | 55.89 | 59.93 |
| s35932 | 71224 | 1763 | 28249 | 160165 | 49802987 | 0.0032 | 0.32 | 5.66 | 6 | 71.54 | 71.54 |
| s38417 | 76678 | 1664 | 27690 | 600879 | 46076160 | 0.0130 | 1.30 | 21.70 | 22 | 87.26 | 91.79 |
| s38584 | 76864 | 1464 | 32841 | 337964 | 48079224 | 0.0070 | 0.70 | 10.29 | 11 | 64.94 | 68.54 |

The distribution of care bit counts for all ISCAS89 benchmark circuits for the transition fault model is shown in Figures 13-22.

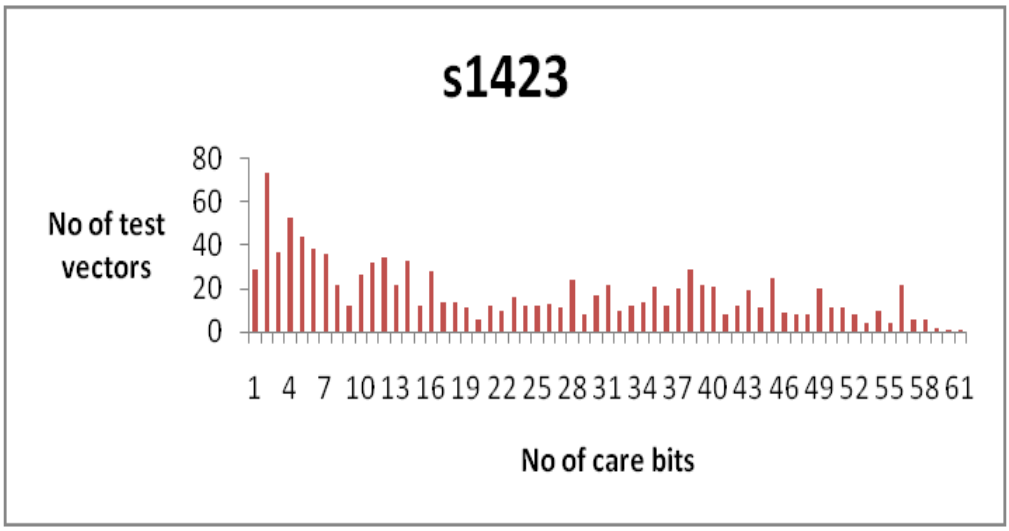


Figure 13. s1423 distribution of care bits

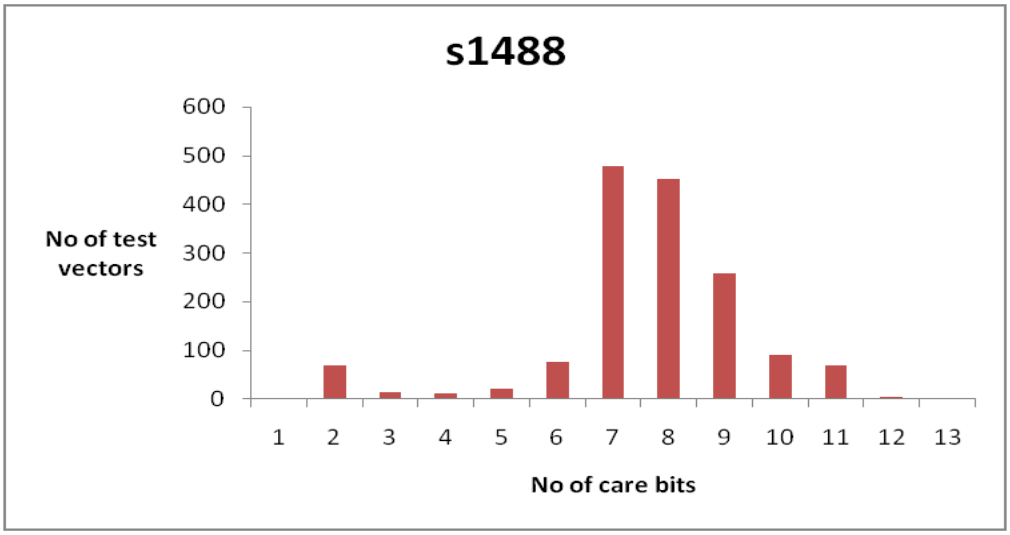


Figure 14. s1488 distribution of care bits

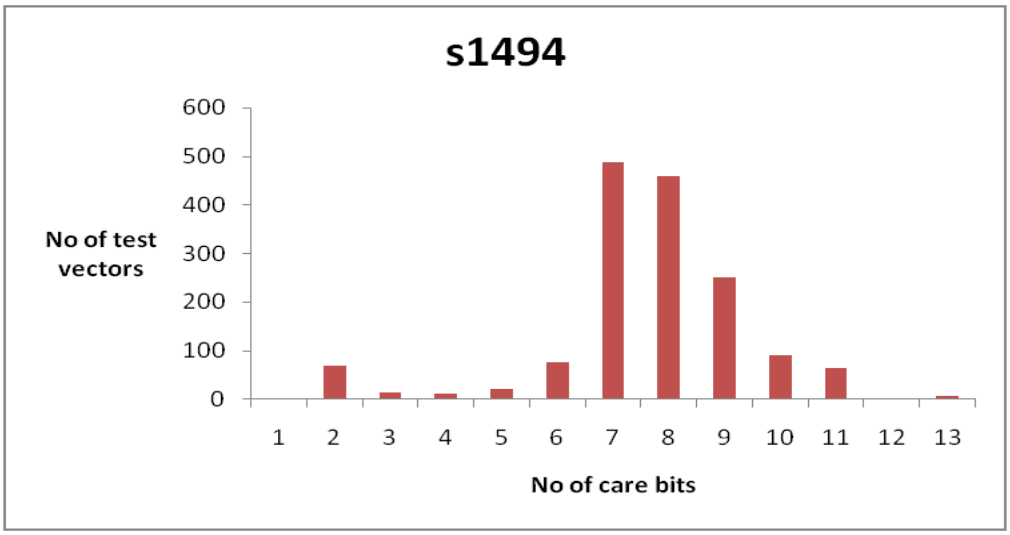


Figure 15. s1494 distribution of care bits

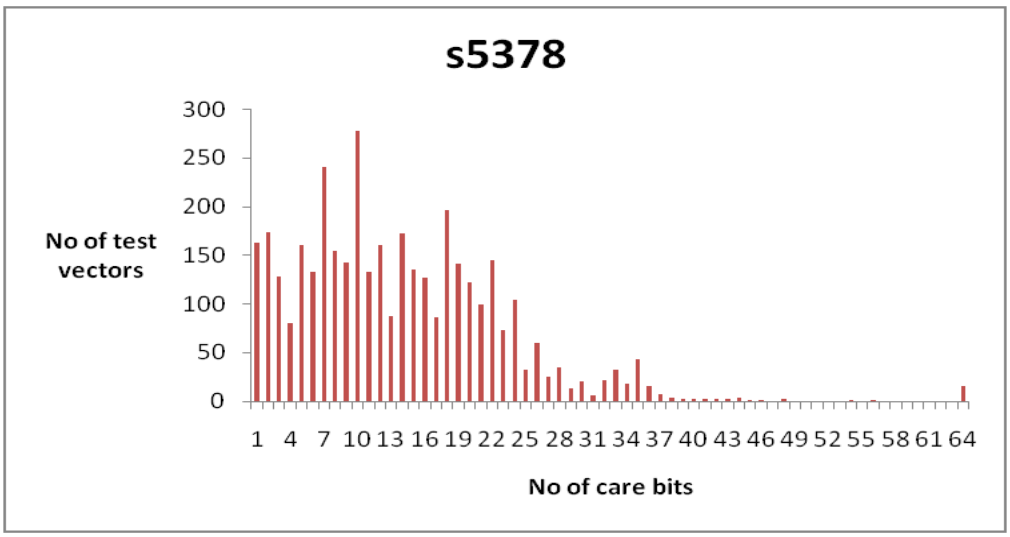


Figure 16. s5378 distribution of care bits

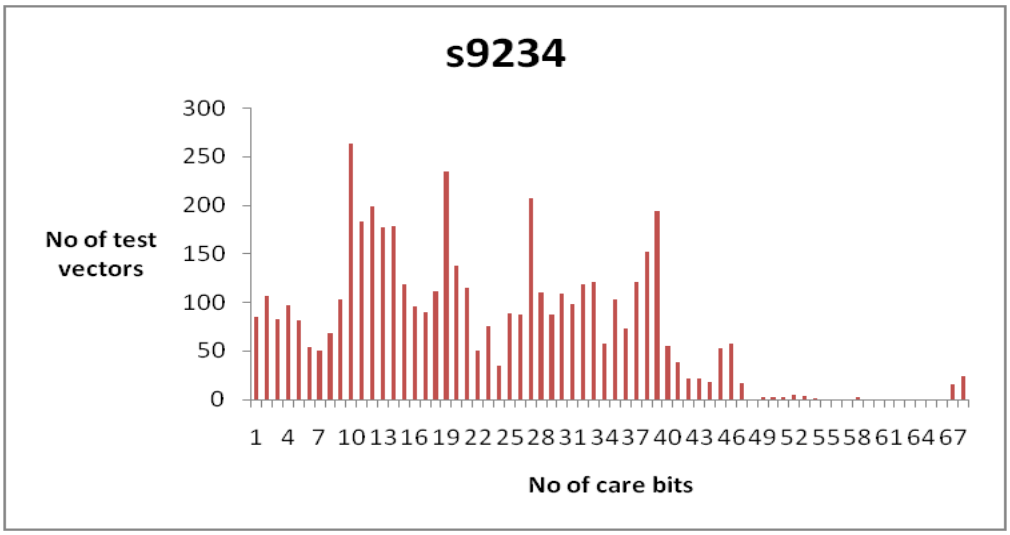


Figure 17. s9234 distribution of care bits

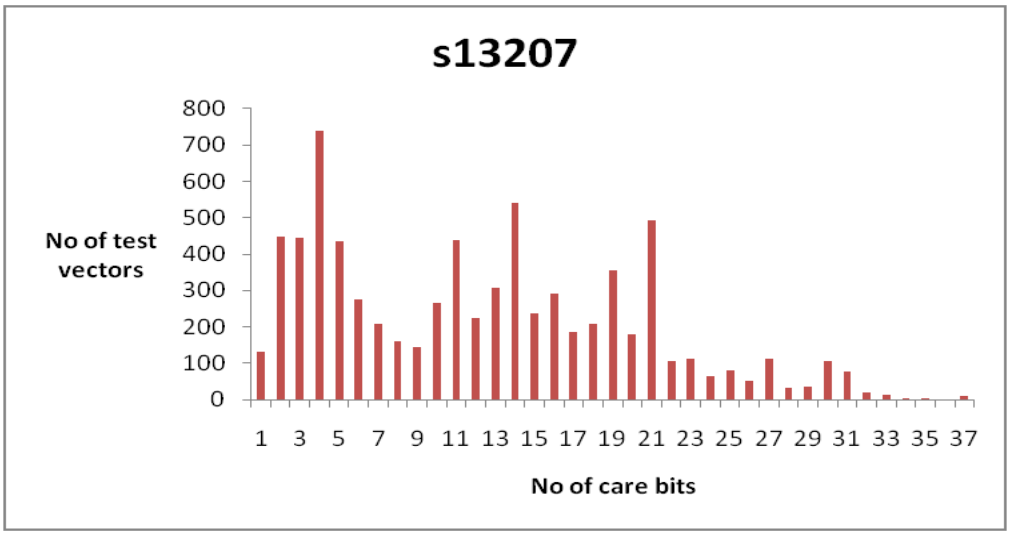


Figure 18. s13207 distribution of care bits

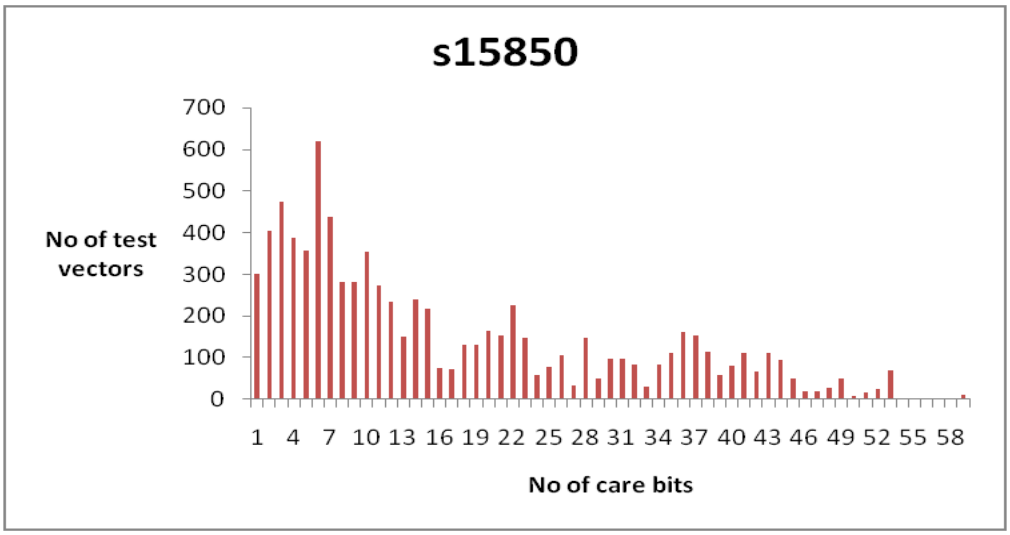


Figure 19. s15850 distribution of care bits

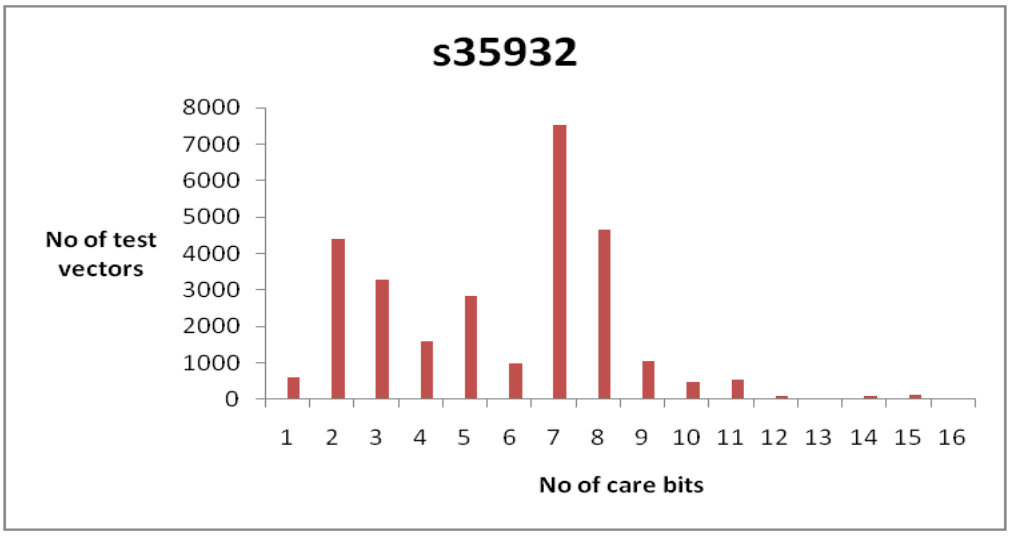


Figure 20. s35932 distribution of care bits

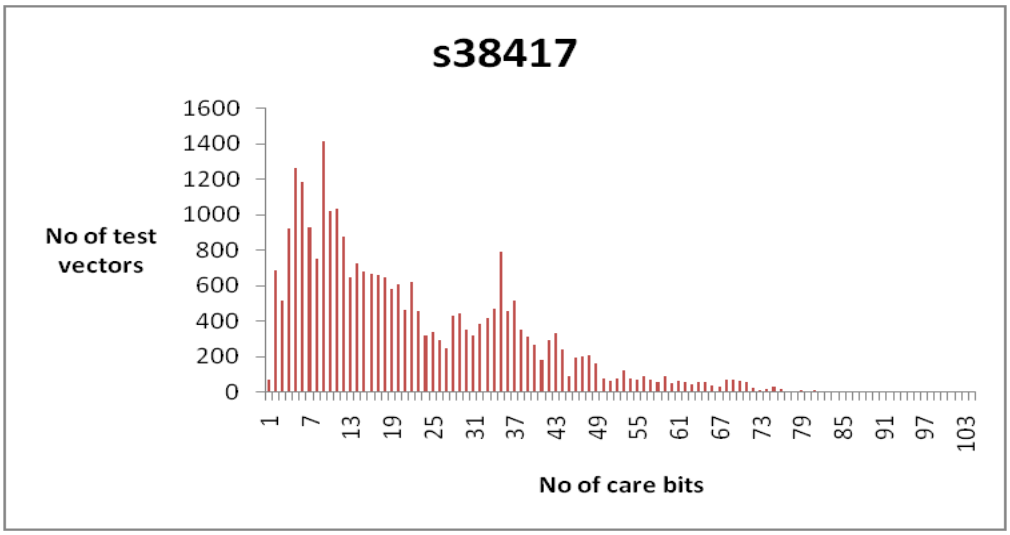


Figure 21. s38417 distribution of care bits

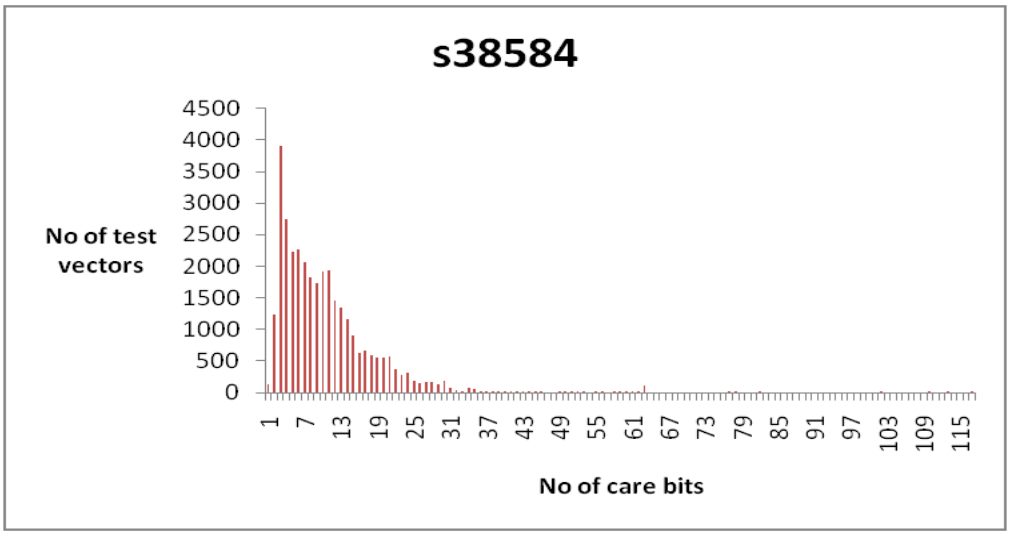


Figure 22. s38584 distribution of care bits

# 5. RESISTIVE OPEN AND BRIDGE FAULT MODELS

## 5.1 Resistive Open Fault Model

An open in the transistors forming a logic gate manifests as a transistor stuck open fault. This can usually be detected by a transition fault test. Opens in wires are not necessarily hard opens i.e., they do not have infinite resistance.

Opens in wires interconnecting logic gates can be classified into strong opens (e.g. > 10MΩ in 180 nm technology) and weak opens (e.g. <= 10MΩ) [30]. Strong opens cause stuck-at faults and therefore, can be detected by regular stuck-at test patterns. Weak opens cause delay faults and therefore require a delay fault test. In the resistive open fault model, a resistive open is represented by a resistor in a net at the location where the open defect may occur as shown in Figure 23 [31]. The open resistance is modeled by an increased delay in the net. The delay increases linearly with the open resistance, as shown in Figure 24 [31]. Above a certain value of open resistance, depending on the clock frequency of the circuit, the open becomes a stuck-open fault. The open resistance distribution is taken from [30] and it is assumed that 80% of the open faults have infinite resistance, while 20% are resistive with *log*(R) uniformly distributed, where R is the open resistance.

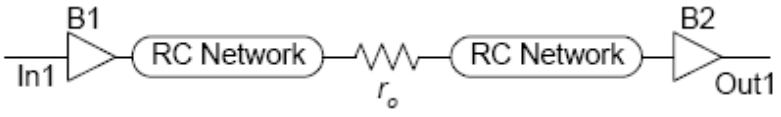


Figure 23. Resistive open fault model [31]

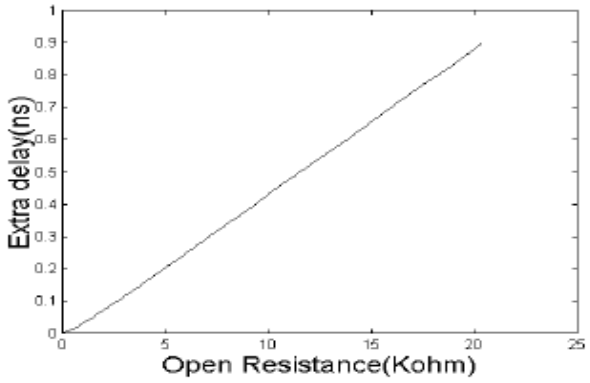


Figure 24. Delay increases linearly with open resistance [31]

The transition fault coverage is the upper bound of resistive open fault coverage because it is only guaranteed to detect large delay increases. The transition fault coverage of a given test pattern set is calculated by determining its effectiveness in generating a transition for every gate in the CUT and propagating the transition to an observation point. That is, transition fault coverage of 100% implies that a transition was generated for every gate in the CUT. In contrast, for the resistive open fault model, the effectiveness of a given test pattern depends on the length of the paths it sensitizes through the fault site. It is important to sensitize long paths because even a small increase in open resistance may cause a delay that exceeds the slack. Hence, the effectiveness of a test set to sensitize long paths can be determined from the resistive open fault coverage of the test set. For resistive open fault coverage, the transition is propagated under robust propagation conditions to an observation point.

The STUMPS-TPS architecture with a single scan chain was used for the following experiments. First, 10,000 unweighted test patterns were applied and the resistive open fault coverage determined. To these 10,000 patterns, an additional 10,000 weighted

random patterns were applied to further improve the coverage. The weighted patterns were weighted using different individual combinational gates such 2-input AND, 2-input NAND, 3-input AND, 3-input NAND, 4-input AND and 4-input NAND gates. An additional experiment was conducted in which to the 10,000 random patterns, an additional 20,000 test patterns were applied. 10,000 of these were obtained by an OA21 weight and the remaining 10,000 were weighted by OAI21. A final experiment was conducted in which to the initial 10,000 random patterns, an additional 80,000 weighted random test patterns were applied, with 10,000 patterns per each of the weights listed in Tables 1 and 2.

Figures 25-32 show the transition fault coverage and resistive open fault coverage obtained for ISCAS89 benchmark circuits. TC (O) is the transition (open) fault coverage. OC (FS) is the resistive open fault coverage under full-speed, i.e. at the rated clock frequency. OC (SS) is the resistive open fault coverage under slow speed, i.e. at half clock frequency. The rated clock period is set to be 5% longer than the delay of the longest structural path. STUMPS-TPS is the coverage for 10,000 pseudo-random TPS patterns. AND2, NAND2, AND3, NAND3, AND4 and NAND4 is the coverage obtained using 10,000 pseudo-random TPS patterns and an additional 10,000 weighted random patterns weighted using these respective combinational gates. Thus, this is the coverage obtained using a total of 20,000 test patterns. OA21 & OAI21 is the coverage obtained using 30,000 test patterns. 10,000 of them were ordinary pseudo-random patterns, 10,000 were weighted using OA21 gate and the remaining 10,000 using OAI21 gate. ALL WEIGHTS is the coverage obtained using 10,000 pseudo-random patterns

and 80,000 weighted random patterns weighted by the eight combinational gates mentioned above. This is the final coverage obtained using all weights.

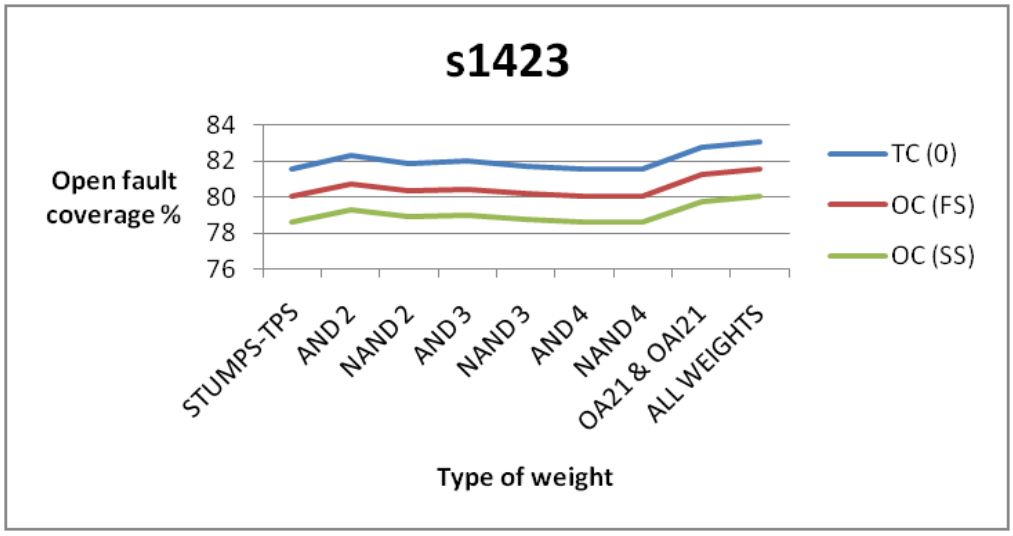


Figure 25. s1423 resistive open coverage for different weights

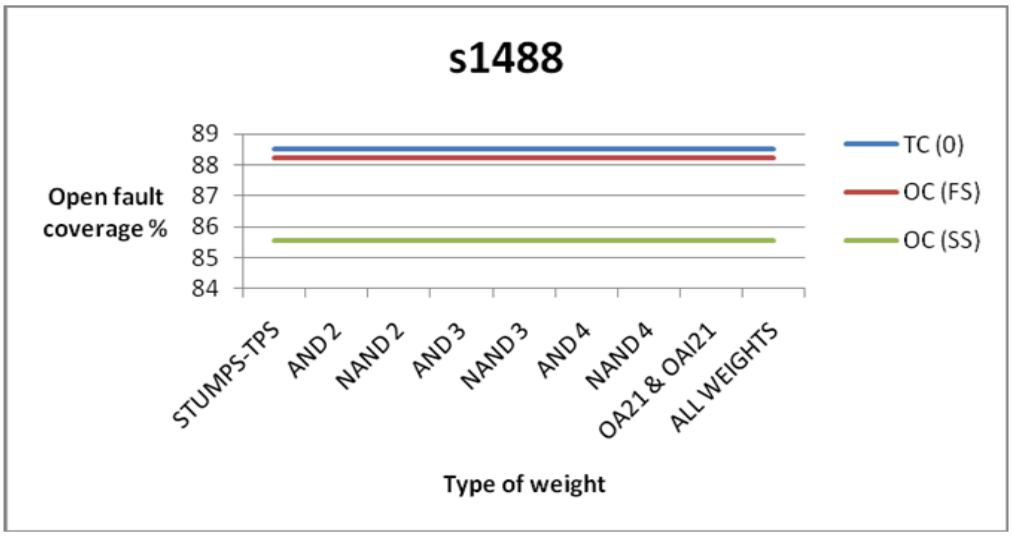


Figure 26. s1488 resistive open coverage for different weights

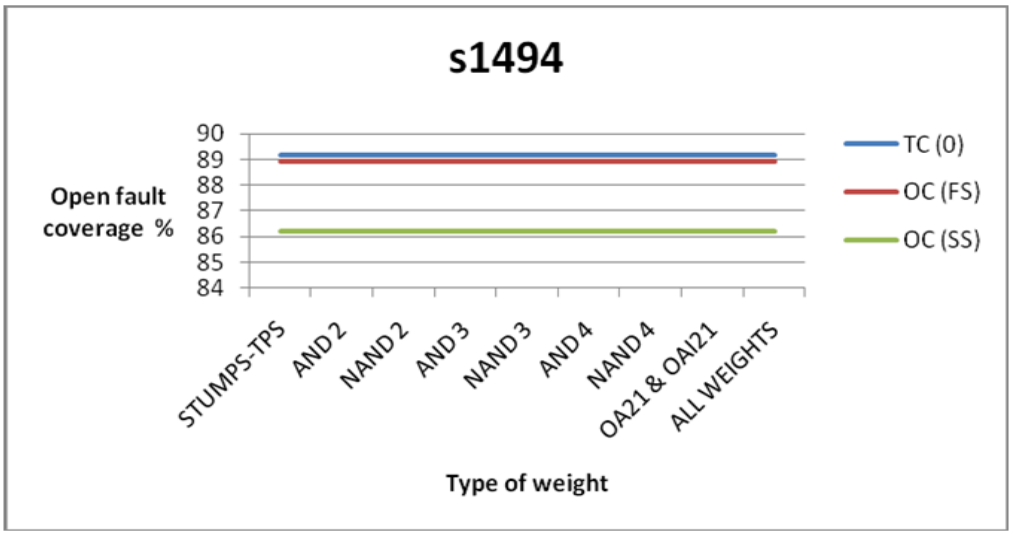


Figure 27. s1494 resistive open coverage for different weights

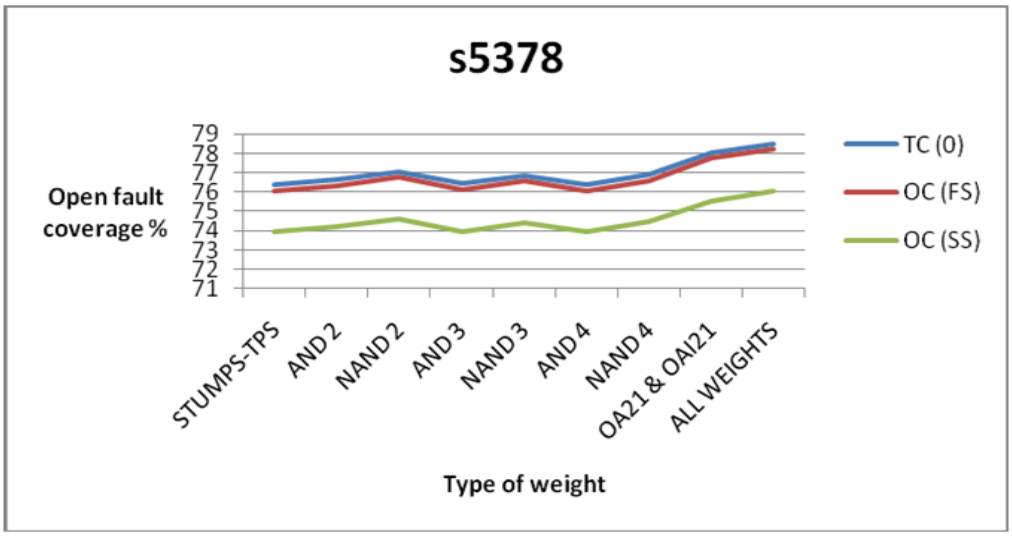


Figure 28. s5378 resistive open coverage for different weights

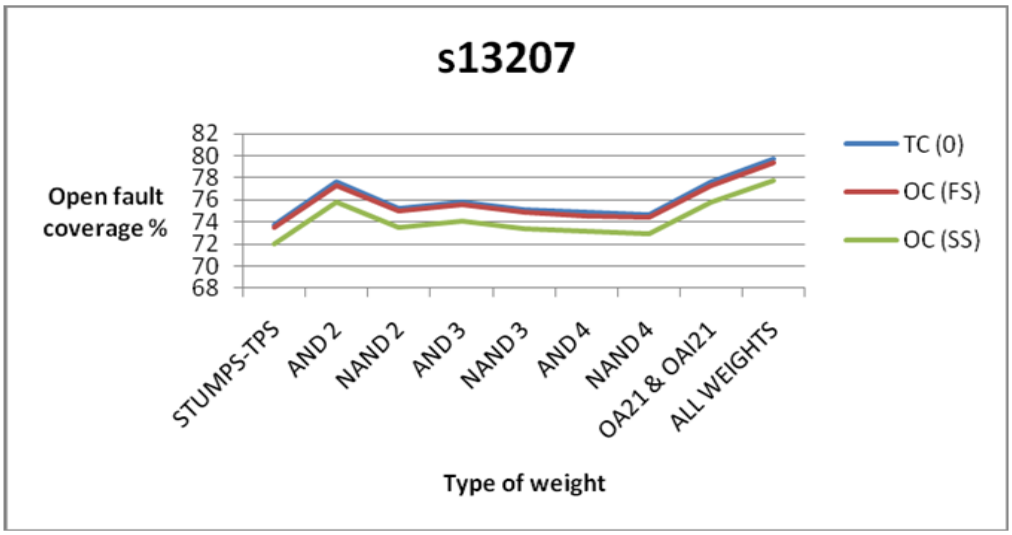


Figure 29. s13207 resistive open coverage for different weights

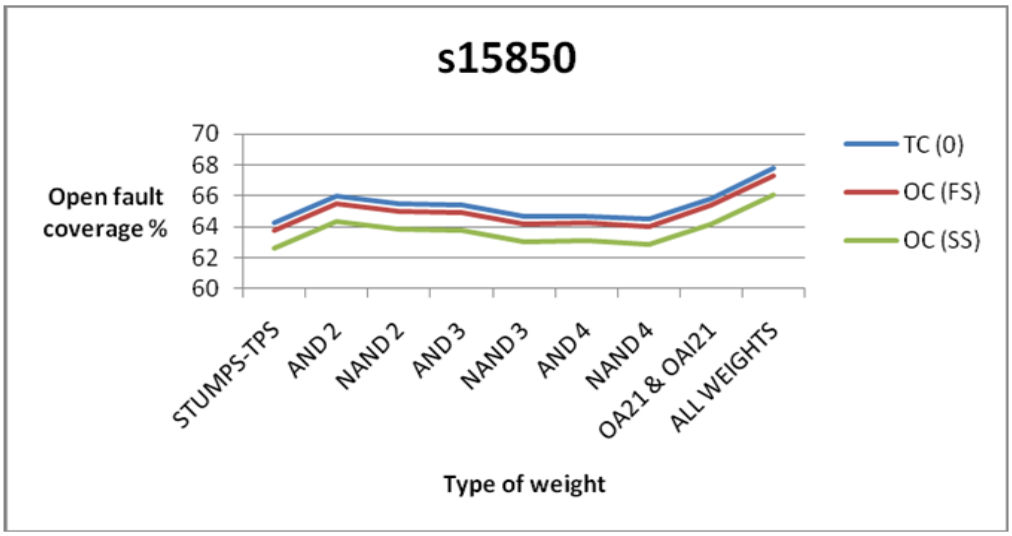


Figure 30. s15850 resistive open coverage for different weights

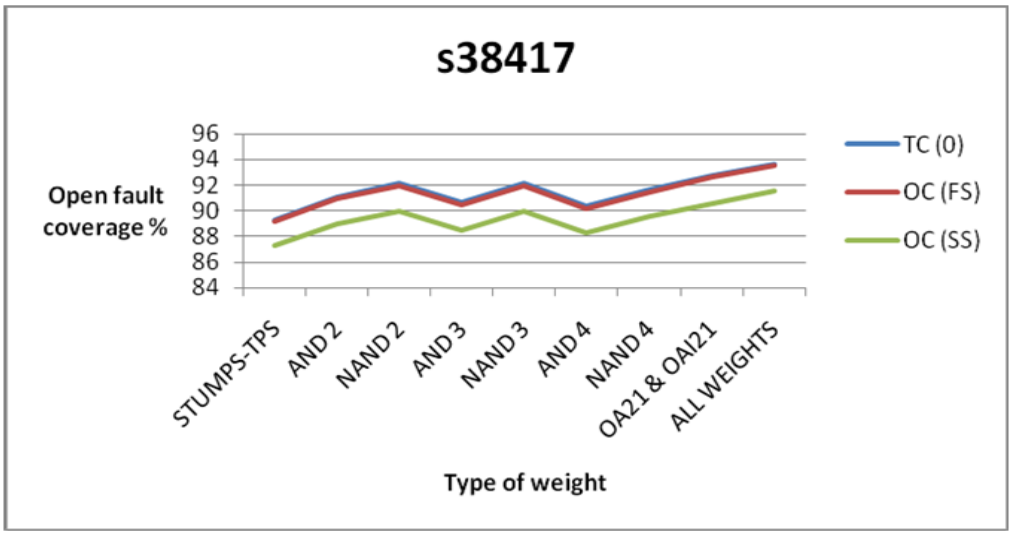


Figure 31. s38417 resistive open coverage for different weights

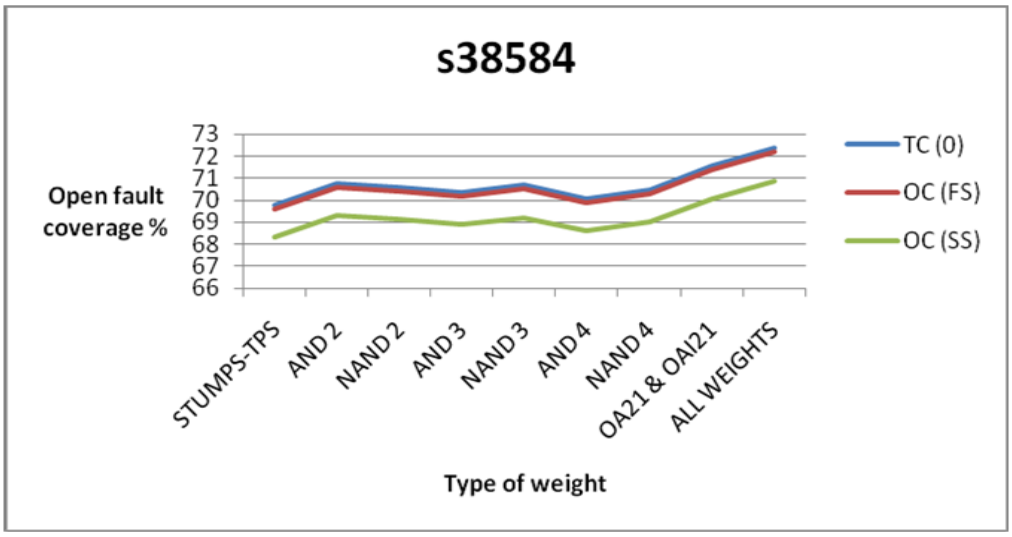


Figure 32. s38584 resistive open coverage for different weights

As can be observed from Figures 25-32, these experiments lead to one important conclusion. There is a strong correlation between transition fault coverage and resistive

open coverage. This is because a drop in transition fault coverage for a given test set was accompanied by a similar drop in resistive open fault coverage. This means that under both these fault models, a transition could not be created for certain lines in the circuit. It can be observed that the resistive open fault coverage is always less than and very close to transition fault coverage. The resistive open fault coverage gives an estimate of how good the given test set is in sensitizing long paths. The transition fault model only checks if a transition is created and propagated. Hence, the transition fault coverage is an upper bound on the resistive open fault coverage. The fact that the resistive open fault coverage follows very closely the transition fault model shows that the transition fault coverage is a good estimator of the obtainable resistive open fault coverage. That is, high transition fault coverage also implies high resistive open fault coverage. As mentioned above, OC (FS) is the resistive open coverage under full-speed and OC (SS) is the resistive open coverage under slow-speed. For ISCAS89 benchmark circuits, it can be observed from Figures 25-32 that there is a significant gap between resistive open coverage under full speed and slow speed. This is because small delay defect detection on long paths is high at full speed, but low at slow speed.

Table 13. WRPG for resistive open fault coverage

| Circuit | STUMPS-TPS | | | BEST WEIGHT | | | | OA21 and OAI21 | | | ALL-WEIGHTS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TC (O) | OC (FS) | OC (SS) | GATE | TC (O) | OC (FS) | OC (SS) | TC (O) | OC (FS) | OC (SS) | TC (O) | OC (FS) | OC (SS) |
| s1423 | 81.58 | 80.02 | 78.58 | AND2 | 82.34 | 80.76 | 79.31 | 82.8 | 81.28 | 79.77 | 83.10 | 81.58 | 80.07 |
| s1488 | 88.51 | 88.23 | 85.57 | NONE | 88.51 | 88.23 | 85.57 | 88.51 | 88.23 | 85.57 | 88.51 | 88.23 | 85.57 |
| s1494 | 89.18 | 88.9 | 86.22 | NONE | 89.18 | 88.9 | 86.22 | 89.18 | 88.9 | 86.22 | 89.18 | 88.9 | 86.22 |
| s5378 | 76.39 | 76.07 | 73.94 | NAND2 | 77.07 | 76.78 | 74.61 | 78.04 | 77.74 | 75.55 | 78.51 | 78.23 | 76.02 |
| s13207 | 73.77 | 73.46 | 72.05 | AND2 | 77.7 | 77.32 | 75.81 | 77.61 | 77.27 | 75.77 | 79.73 | 79.37 | 77.8 |
| s15850 | 64.23 | 63.76 | 62.6 | AND2 | 66.01 | 65.52 | 64.33 | 65.86 | 65.37 | 64.17 | 67.77 | 67.28 | 66.04 |
| s38417 | 89.31 | 89.16 | 87.25 | NAND2 | 92.11 | 91.97 | 89.99 | 92.74 | 92.61 | 90.6 | 93.68 | 93.56 | 91.52 |
| s38584 | 69.78 | 69.6 | 68.34 | AND2 | 70.78 | 70.6 | 69.31 | 71.58 | 71.4 | 70.09 | 72.37 | 72.19 | 70.87 |

Examination of Table 13 gives the following results. Column 5 gives the best weight that achieves highest improvement in fault coverage using 10,000 weighted random vectors topped over the 10,000 pseudo-random vectors. 2-input AND or 2-input NAND gate gives the highest improvement in fault coverage. 2-input AND gate gives a 0.75 probability for logic 0 and a 0.25 probability for logic 1. Similarly, 2-input NAND gate gives a 0.75 probability for logic 1 and a 0.25 probability for logic 0. These gates bias the output probability of an LFSR from 0.5 to either 0.25 or 0.75. Increasing the number of inputs to the AND or NAND gate increases the bias of the output. For example, a 3-input AND gate has a 0.875 probability for logic 0 and a 0.125 probability for logic 1. This probability closer to 0 increases the number of zeros that are fed into the scan chain. This results in the test vector being filled with too many zeros and hence decreases the probability of detecting a fault. On the other hand, 2-input AND gate biases the

probability only slightly to 0.75 and hence there is still a significant number of ones in the test vector. The same argument holds for a 2-input NAND gate. Thus, these results prove that high biasing of the output of the LFSR results in less improvement in fault coverage. Hence, it can be concluded that 2-input gates give better improvement in fault coverage over gates that have more inputs. This is because there is still a good distribution of both zeros and ones in the generated test vector.

The fault coverage obtained using both OA21 and OAI21 are given in Columns 9, 10 and 11. These show that using these two weights gives a better improvement in fault coverage than using a single weight. These two weights were again chosen using the conclusion derived from observing Column 5 that a slight bias of the probability gives a better improvement in fault coverage over a large bias. Hence, gates OA21 and OAI21 which provide a slight bias were used. For example OA21 biases logic 0 to have a probability of 0.625 and logic 1 to have a probability of 0.375. These are only slightly away the usual probability of 0.5 for an ordinary LFSR. Similar is the insight for using OAI21. Columns 12, 13 and 14 give the final fault coverage obtained using all weights. Using all the weights gives the highest fault coverage. Resistive open results obtained are pessimistic because the circuit clock period was set based on the delay of the longest structural path. That is, the length of the longest path is set to that of the longest structural path. However, the true longest path (the longest sensitizable path) can be smaller than the longest structural path. They are equal only when there are no false paths, i.e., all paths are functionally sensitizable.

## 5.2 Resistive Bridge Fault Model

The resistive bridge fault model is shown in Figure 33 [31].

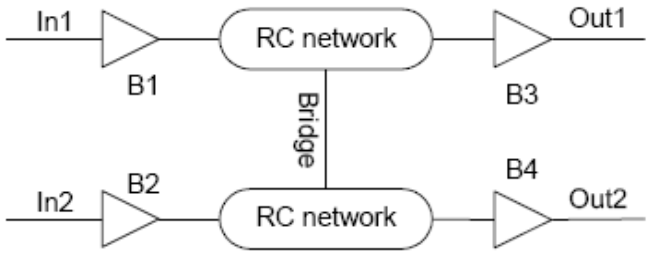


Figure 33. Resistive bridge fault model [31]

Resistive bridges can cause an increase or decrease in delay, depending on the voltage of the bridged node. In this work we only consider delay increases. To test a short between two lines, the test pattern must generate a transition on one or both of the lines. If the initial and final values are the same in both lines, then no transition is created and the short between the two lines does not cause any increase in delay. On the other hand, if a transition is generated on one or both lines and the final values of both lines are different, then the short between the two lines may cause an increase in delay. Hence, the objective of the test vector pair is to generate transition on one or both lines and the final values of both lines must be different. If there is a transition on only one line, the other line has a stable logic value and hence the line with the transition may be subject to a delay and is the activating transition. If there are transitions on both lines, the line with the latest transition determines the delay due to this short and is the activating transition. An additional objective of the test vector is to propagate the

activating transition under robust propagation conditions to an observation point namely a primary output.

Four different cases of transitions can be subjected to increase in delay due to a resistive short between two nets Net1 and Net2. They are 1) Net1 rising 2) Net1 falling 3) Net2 rising and 4) Net2 falling. The particular case for a test vector pair is determined by the activating transition. The transition fault coverage of a given test set tells us if a transition was created and propagated to an observation point or not. However, it does not take into account the length of the path through which the transition is propagated. The resistive bridge fault coverage gives us an estimate of the length of the path through which the transition is propagated under robust propagation conditions. This is because resistive bridge coverage determines the maximum detected resistance for every bridge in the CUT. The detected bridge resistance is maximized when the length of the robust path through which the activating transition is propagated is also maximized. Hence, the resistive bridge coverage determines the effectiveness of a given test set in propagating transition through the longest path for a gate driving the line which is shorted with another line (for the bridge under test). The longer the path through which the transition is propagated, the smaller the slack. With a smaller slack, even a small increase in delay might be sufficient to fail timing. The larger the increase in bridge resistance, the smaller the delay increase. Hence, it is necessary to propagate the transition through the longest paths (which have the smallest slacks) so that even the smallest increase in delay (caused by the largest bridge resistance) can be detected. Hence, resistive bridge coverage

determines the largest detectable resistance of all bridges which is determined by the length of the robust path through which the transition has been propagated.

The STUMPS architecture has been used for the experiments. Again TPS vectors were generated and fault coverage was determined. To improve this fault coverage, weighted random patterns weighted by 10 different weights were used. The combinational gates used for weighting are 2-input AND, NAND, OR and NOR gates, 3-input AND and NAND gates, 4-input AND and NAND gates and OA21 and OAI21 gates. As a final experiment, the effectiveness of all the weights put together in improving fault coverage was determined. These experiments were evaluated on ISCAS85 benchmark circuits [32]. ISCAS85 circuits have been used rather than ISCAS89 circuits because the ISCAS85 benchmarks had pull-up/pull-down resistances of all gates available. Random non-feedback shorts were used. The number of shorts is approximately twice the number of lines in the circuits. Shorts between lines feeding the same gate are not included. Shorts between the signal lines and power/ground grid are not considered because they are most likely to behave as stuck-at or transition faults. The bridge resistance is assumed to be uniformly distributed between 0Ω and 40kΩ [33].

The results of these experiments on resistive bridge faults are shown in Figures 34-41.

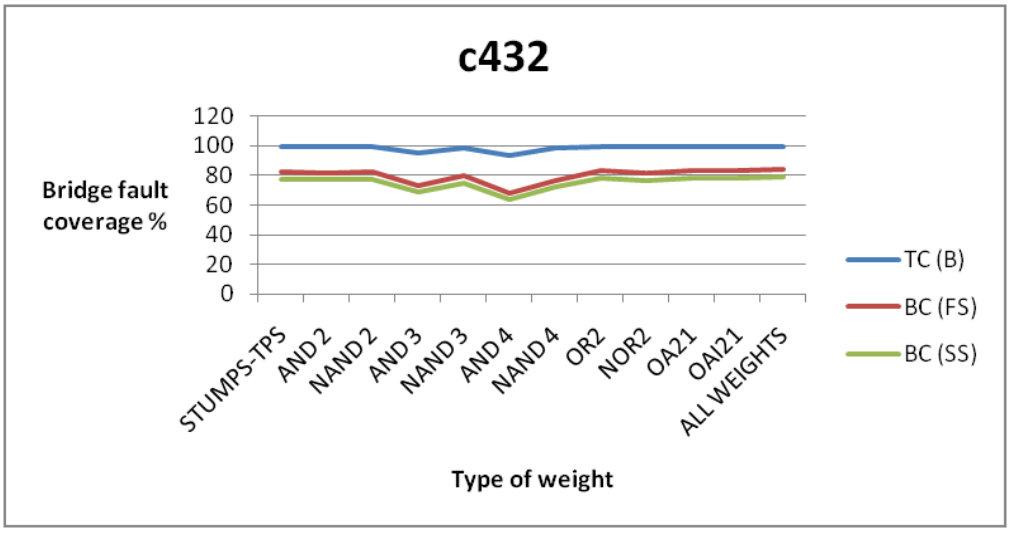


Figure 34. c432 resistive bridge coverage for different weights

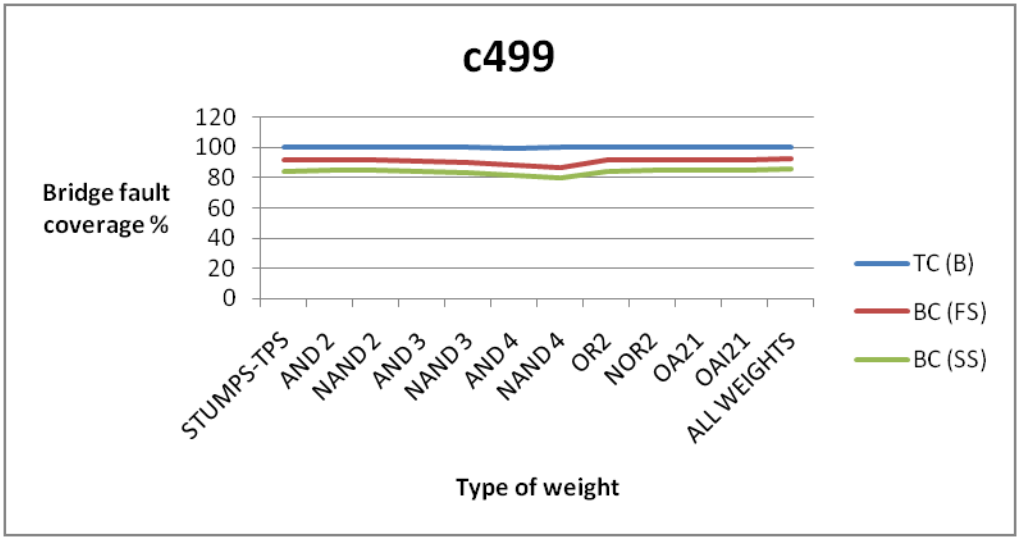


Figure 35. c499 resistive bridge coverage for different weights

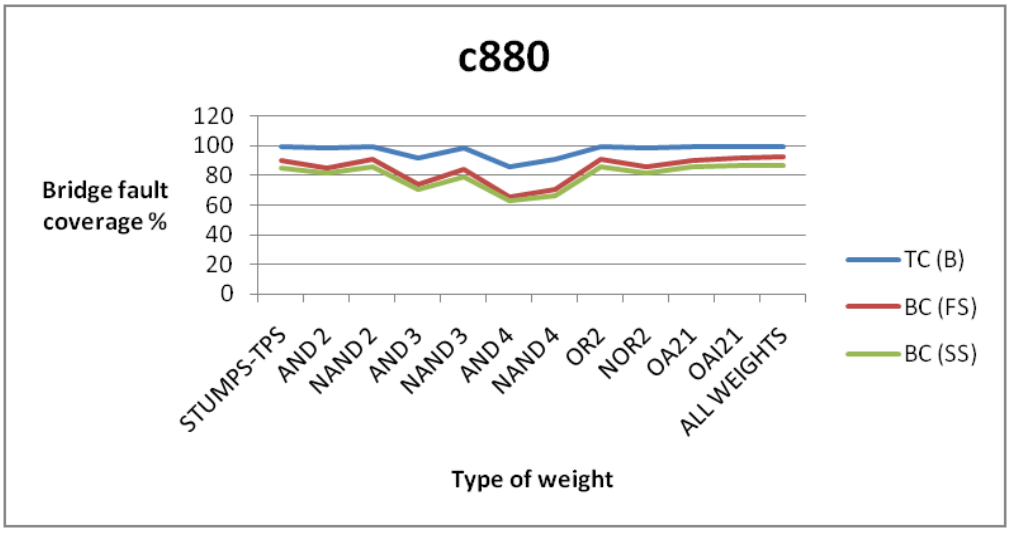


Figure 36. c880 resistive bridge coverage for different weights

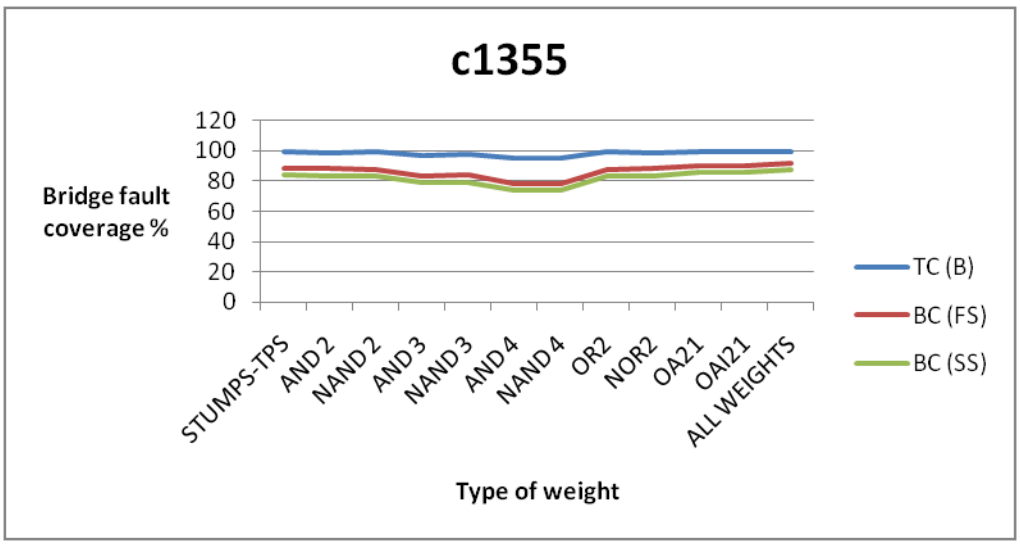


Figure 37. c1355 resistive bridge coverage for different weights

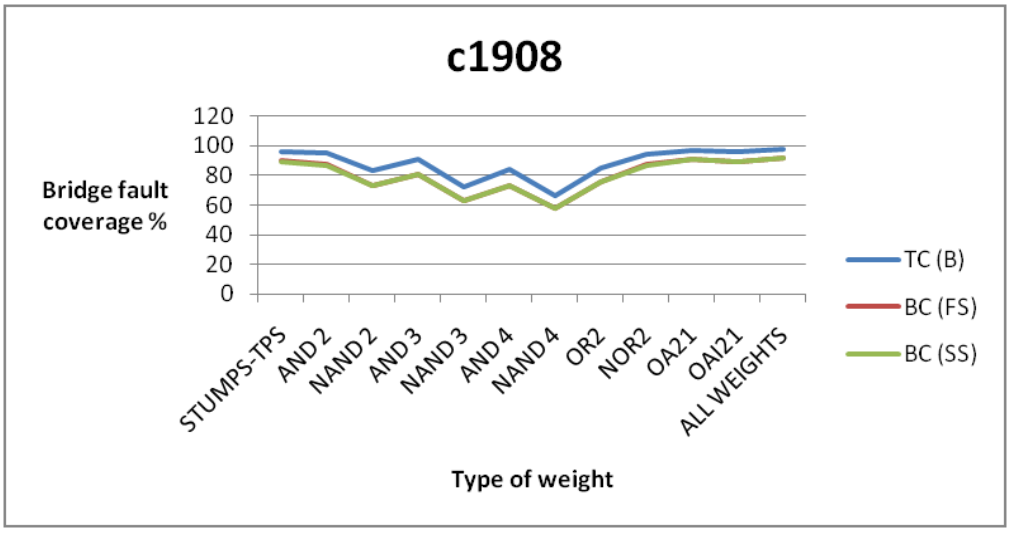


Figure 38. c1908 resistive bridge coverage for different weights

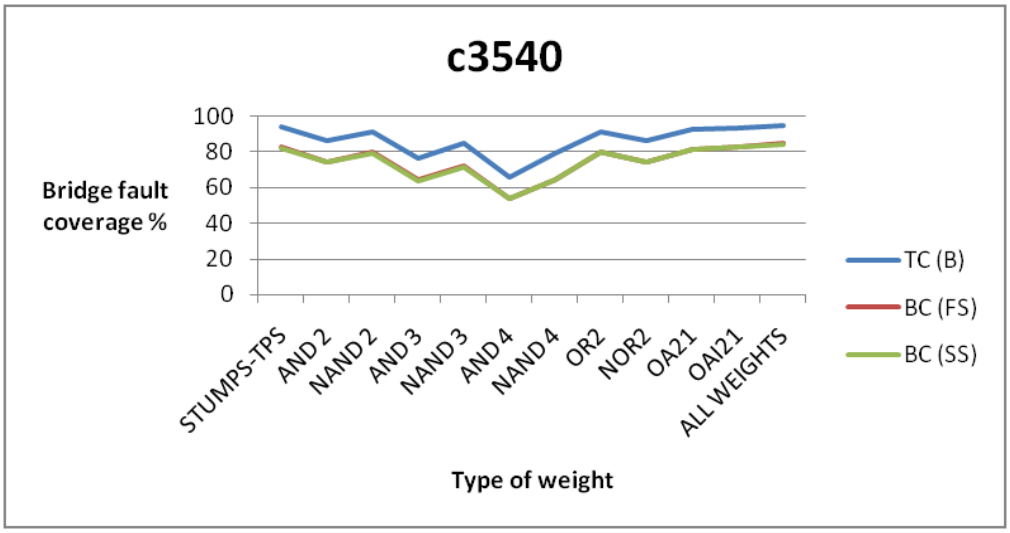


Figure 39. c3540 resistive bridge coverage for different weights

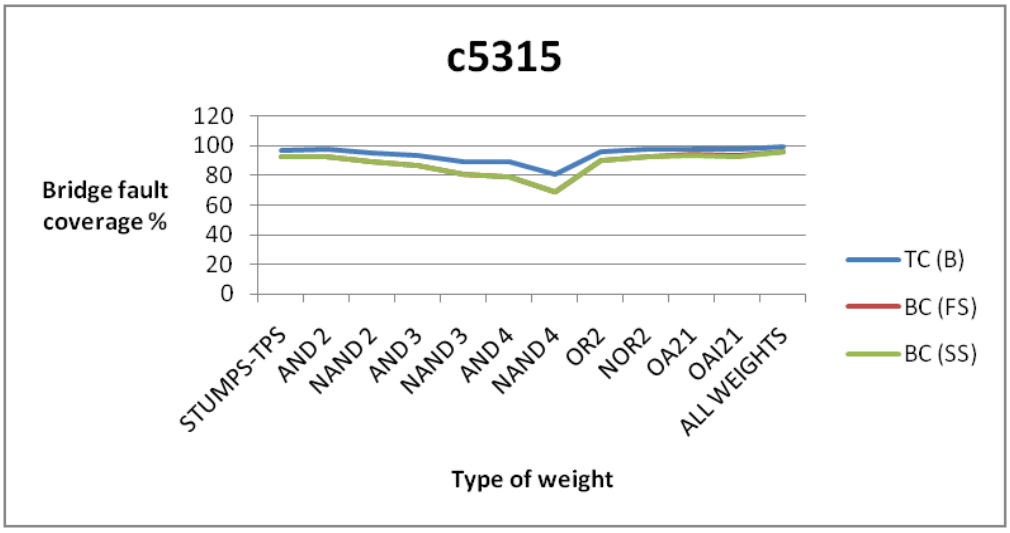


Figure 40. c5315 resistive bridge coverage for different weights

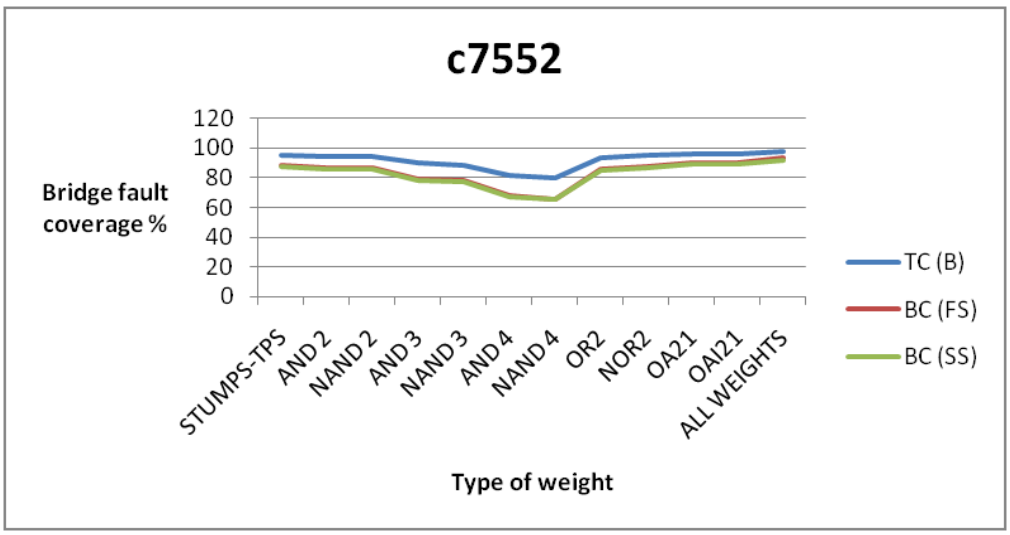


Figure 41. c7552 resistive bridge coverage for different weights

Figures 34-41 show a strong correlation between transition fault coverage and resistive bridge fault coverage. This again indicates that high transition fault coverage

implies high resistive bridge fault coverage. The impact of using different weights on resistive bridge coverage is similar to that in resistive open coverage and has already been explained. Use of all weights gives the highest fault coverage under both models. BC (FS) is the resistive bridge coverage under full-speed. BC (SS) is the resistive bridge coverage under slow-speed. For ISCAS85 benchmark circuits, it can be observed from Figures 34-41 that the resistive bridge coverage under slow speed is very close to the resistive bridge coverage under full speed. This is because the clock period is set by a few long paths, so most sensitized paths have a large slack, even at full speed.

Table 14. WRPG for resistive bridge fault coverage

| CIRCUIT | STUMPS-TPS | | | BEST WEIGHT | | | | ALL-WEIGHTS | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | TC (B) | BC (FS) | BC (SS) | GATE | TC (B) | BC (FS) | BC (SS) | TC (B) | BC (FS) | BC (SS) |
| c432 | 99.26 | 82.23 | 77.34 | OA21 | 99.26 | 83.56 | 78.15 | 99.26 | 84.39 | 78.85 |
| c499 | 100 | 91.34 | 84.43 | OA21 | 100 | 91.9 | 84.96 | 100 | 92.49 | 85.53 |
| c880 | 99.43 | 89.8 | 85.22 | OAI21 | 99.64 | 91.97 | 86.88 | 99.64 | 92.34 | 87.06 |
| c1355 | 99.27 | 88.3 | 83.85 | OAI21 | 99.51 | 90.36 | 85.77 | 99.59 | 92.01 | 87.28 |
| c1908 | 96.08 | 89.8 | 89.54 | OA21 | 96.63 | 90.81 | 90.53 | 97.37 | 91.93 | 91.64 |
| c3540 | 93.86 | 82.56 | 82.3 | OAI21 | 93.76 | 82.96 | 82.7 | 94.84 | 84.71 | 84.45 |
| c5315 | 97.22 | 92.6 | 92.32 | OA21 | 97.77 | 94.08 | 93.79 | 99 | 96.06 | 95.76 |
| c7552 | 94.89 | 88.1 | 87.25 | OAI21 | 95.75 | 90.01 | 89.15 | 97.36 | 93.04 | 92.18 |

Analysis of Table 14 shows that either OA21 or OAI21 achieved the highest improvement in fault coverage for all ISCAS85 benchmark circuits. This proves that a slight bias of the outputs of an LFSR provides higher improvement in fault coverage over using weights which provides a large bias. That is, weights OA21 and OAI21 with probabilities 0.625 and 0.375 (which provide a slight bias from 0.5) are more effective in improving fault coverage than weights AND3, NAND3, AND4, NAND4 etc (which provide a large bias from 0.5). This is a significant result in designing weighting schemes. These weights (OA21 or OAI21) provide the highest improvement in resistive bridge coverage when a single weight is used. The slight bias provided by OA21 or OAI21 weights prevents filling of scan chain with excessive zeros or ones and therefore achieves the highest fault coverage. Similarly, a combination of all weights also provides the highest achievable transition and bridge fault coverage.

# 6. CONCLUSIONS AND FUTURE WORK

## 6.1 Conclusions

Three BIST approaches namely BILBO, STUMPS-Test Per Clock (TPC) and STUMPS-Test Per Scan (TPS) were evaluated. It was found that STUMPS-TPS provides the highest delay fault coverage with the least impact on existing scan design. This STUMPS-TPS approach was used in all future experiments. Experiments were carried out to evaluate different weighted random pattern generation (WRPG) techniques by varying the weight. The experiment to find the best weight showed that 2-input gates were more effective than higher-input gates in improving transition fault coverage. It was found that hitherto unexplored gates like OA21 and OAI21 which biased the output of the LFSR only slightly provided a better improvement in realistic fault coverage over usual gates like AND and NAND. During our experiments, it was also found that the type of combinational gate used to realize a given probability determines the improvement in fault coverage obtained. Use of care bit density of ATPG test vector set to determine effectiveness of WRPG techniques was also studied. The effectiveness of WRPG in sensitizing long paths was evaluated using resistive open and resistive short fault models. It was found that the transition fault coverage was a good estimate (indicator) of the obtainable resistive open and resistive bridge fault coverage.

## 6.2 Future Work

As explained above, transition fault coverage for opens tells us if an open was detected or not. Resistive open fault coverage gives an estimate of how good the given test set is in sensitizing long paths. Hence, to improve transition and resistive open

coverage, new opens must be detected. This can be achieved by inserting test points. However, test points cannot be added to critical paths (which have small slacks) because the added points themselves increase the delay of the critical path thus reducing the slack further. Hence, addition of test points on non-critical paths is a future topic of interest. This can boost both transition and resistive open coverage by detecting new opens. The added test points would also improve the resistive bridge fault coverage by detecting new bridges.

# REFERENCES

[1] L-T. Wang, C.E. Stroud and N.A. Touba, *System-on-chip Test Architectures Nanometer Design for Testability*, Burlington, MA: Morgan Kaufmann, 2008.

[2] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K-H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide and J. Qian, "Embedded deterministic test for low cost manufacturing test," in *Proc. Int. Test Conf.*, Baltimore, MD, 2002, pp. 301-310.

[3] H.J. Wunderlich, "BIST for Systems-on-a-Chip," *Integration, the VLSI Journal*, vol. 26, no. 1-2, pp.55-78, Dec. 1998.

[4] ITRS, "International Technology Roadmap for Semiconductor," Tech. Rep., 1999 [Online]. Available: http://www.itrs.net/links/1999Summer/SantaClara1999.html

[5] A.K. Majhi and V.D. Agrawal, "Delay fault models and coverage," in *Proc. Int. Conf. VLSI Design*, Chennai, India, 1998, pp. 364-369.

[6] S. Patil and J. Savir, "Skewed-load transition test: Part II, Coverage," in *Proc. Int. Test Conf.*, Baltimore, MD, 1992, pp. 714-722.

[7] J. Savir and S. Patil, "On broad-side delay test," *IEEE Trans. VLSI Syst.*, vol. 2, no. 3, pp. 368-372, Sep. 1994.

[8] G. Xu and A.D. Singh, "Low cost Launch-on-shift delay test with slow scan enable," in *Proc. Eur. Test Symp.*, Southampton, UK, 2006, pp. 9-14.

[9] G. Xu and A.D. Singh, "Delay test scan flip-flop: DFT for high coverage delay testing," in *Proc. Int. Conf. VLSI Design*, Bangalore, India, 2007, pp. 763-768.

[10] W. Wang and S.K. Gupta, "Weighted random robust path delay testing of synthesized multilevel circuits," in *Proc. IEEE VLSI Test Symp.*, Cherry Hill, NJ, 1994, pp. 291-297.

[11] P. Girard, C. Landrault, V. Mor´eda and S. Pravossoudovitch, "An optimized BIST test pattern generator for delay testing," in *Proc. IEEE VLSI Test Symp.*, Monterey, CA, 1997, pp. 94–99.

[12] B. Dervisoglu and G. Stong, "Design for testability: using scan path techniques for path-delay test and measurement," in *Proc. Int. Test Conf.*, Nashville, TN, October 1991, pp. 365–374.

[13] H.D. Schnurmann, E. Lindbloom and R.G. Carpenter, "The weighted random test-pattern generator," *IEEE Trans. Computers*, vol. 24, no. 7, pp. 695-700, Jul. 1975.

[14] J.A. Waicukauski, E. Lindbloom, E.B. Eichelberger and O.P. Forlenza, "WRP: A method for generating weighted random test patterns," *IBM J. Res. Dev.*, vol. 33, no. 2, pp. 149-161, Mar. 1989.

[15] J. Hartmann and G. Kemnitz, "How to do weighted random testing for BIST," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, Santa Clara, CA, 1993, pp. 568-571.

[16] S. Wang, "Low hardware overhead scan based 3-weight weighted random BIST," in *Proc. Int. Test Conf.*, Baltimore, MD, 2001, pp. 868-877.

[17] N.Z. Basturkmen, S.M. Reddy and I. Pomeranz, "Pseudo random patterns using Markov sources for scan BIST," in *Proc. Int. Test Conf.*, Baltimore, MD, 2002, pp. 1013-1021.

[18] Y. Zorian and A. Ivanov, "EEODM: An effective BIST scheme for ROMs," in *Proc. Int. Test Conf.,* Washington, D.C., 1990, pp. 871-879.

[19] L. Chen, Z. Wen, Z. Zhang and W. Min, "A novel BIST approach for testing input / output buffers in SOCs," in *Proc. Int. Conf. Testing and Diagnosis*, Chengdu, China, 2009, pp. 1-3.

[20] E.J. McCluskey, "Built-in self test techniques," *IEEE Design and Test of Computers*, vol. 2, no. 2, pp. 21-28, Apr. 1985.

[21] L-T. Wang, C-W. Wu and X. Wen, *VLSI Test Principles and Architectures Design for Testability*, San Francisco, CA: Morgan Kaufmann, 2006.

[22] B. Konemann, J. Mucha and G. Zwiehoff, "Built-in test for complex digital integrated circuits," *IEEE J. Solid State Circuits*, vol. 15, no. 3, pp. 315-319, Jun. 1980.

[23] P.H. Bardell and W.H. McAnney, "Self-testing of multiple logic modules," in *Proc. Int. Test Conf.*, Philadelphia, PA, 1982, pp. 200-204.

[24] Y-C. Lin, F. Lu and K-T. Cheng, "Pseudo-functional scan-based BIST for delay fault," in *Proc. IEEE VLSI Test Symp.,* Palm Springs, CA, 2005, pp. 229-234.

[25] C.E. Stroud, "A Designer's Guide to Built-in Self Test," New York: Kluwer Academic Publishers, 2002.

[26] W. Qiu, J. Wang, D.M.H. Walker, D. Reddy, Z. Li, W. Shi and H. Balachandran, "K Longest Paths per Gate (KLPG) test generation for scan-based sequential circuits," in *Proc. Int. Test Conf*., Charlotte, NC, 2004, pp. 223-231.

[27] J.A. Waicukauski, E. Lindbloom, B. Rosen and V. Iyengar, "Transition fault simulation by parallel single fault propagation," in *Proc. Int. Test Conf.*, Washington, D.C., 1986, pp. 542-549.

[28] F. Brglez, D. Bryan and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, Portland, OR, 1989, pp. 1929-1934.

[29] W. Qiu, Z. Li, X. Lu, W. Shi and D.M.H. Walker, "CodSim – A fault simulator for combined delay faults," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems (DFT)*, Boston, MA, 2003, pp. 79-86.

[30] R.R. Montanes, J.P. de Gyvez and P. Volf, "Resistance characterization for weak open defects," *IEEE Design and Test of Computers*, vol. 19, no. 5, pp. 18-26, Sept./Oct. 2002.

[31] Z. Li, X. Lu, W. Qiu, W. Shi and D.M.H. Walker, "A circuit level fault model for resistive opens and bridges," in *Proc. IEEE VLSI Test Symp.*, Napa Valley, CA, 2003, pp. 379-384.

[32] F. Brglez and H. Fujiwara, "A neural netlist of 10 combinational benchmark circuits and a target translator in FORTRAN," in *Proc. Int. Symp. Circuits and Systems*, Kyoto, Japan, 1985, pp. 663-698.

[33] M. Spica, M. Tripp and R. Roeder, "A new understanding of bridge defect resistances and process interactions from correlating inductive fault analysis predictions

to empirical test results," in *Int. Workshop Defect Based Testing*, Marina Del Rey, CA, 2001, pp. 11-16.

# VITA

Karthik Prabhu Tamilarasan received his Bachelor of Engineering degree in Electronics and Communication Engineering from Sri Venkateswara College of Engineering, Sriperumbudur affiliated to Anna University, Chennai in 2008. He entered the Department of Electrical and Computer Engineering at Texas A&M University in August 2008 and received his Master of Science degree in December 2010. His research interests include Design for Testability, Computer Architecture and Design Verification.

Karthik may be reached at the Department of Computer Science and Engineering, 515B HRBB, Texas A&M University, College Station, Texas, 77843. His email is tkptamu@neo.tamu.edu.